TRƯỜNG ĐẠI HỌC KHOA HỌC HUẾ

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TIỂU LUẬN

MÔN: PHÁT TRIỂN ỨNG DỤNG IOT

NHÓM: 4

Đề tài: Hệ thống giám sát độ rung với ESP32

HUẾ, 04/2025

LỜI CẢM ƠN

Trước hết, em xin gửi lời cảm ơn sâu sắc đến quý thầy cô trong Khoa Công nghệ Thông tin, đặc biệt là giảng viên bộ môn Phát triển ứng dụng IoT – người đã tận tình giảng dạy và hướng dẫn em trong suốt quá trình học tập và thực hiện đề tài. Những kiến thức quý báu cùng sự chỉ bảo tận tâm của thầy/cô đã giúp em hiểu rõ hơn về các ứng dụng thực tiễn của công nghệ IoT, đồng thời là nền tảng quan trọng để em hoàn thành tiểu luận này.

Em cũng xin gửi lời cảm ơn chân thành đến các bạn học trong lớp, những người đã không ngại chia sẻ kiến thức, kinh nghiệm và hỗ trợ em trong suốt quá trình tìm hiểu và xây dựng mô hình. Sự hợp tác, góp ý và khích lệ từ các bạn là nguồn động lực quý giá để em hoàn thiện đề tài một cách tốt nhất.

Dù đã nỗ lực và cố gắng trong quá trình thực hiện, bài tiểu luận chắc chắn không tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm, góp ý và phản hồi từ thầy cô cũng như các bạn để đề tài được hoàn thiện hơn trong tương lai.

Trân trọng.

Em xin chân thành cảm ơn !

MỤC LỤC

PHẦN MỞ ĐẦU

1. Lý do chọn đề tài

Trong lĩnh vực công nghiệp hiện đại, các thiết bị và máy móc thường phải hoạt động liên tục trong điều kiện khắc nghiệt, dẫn đến hiện tượng rung động không mong muốn. Nếu không được phát hiện kịp thời, các rung động bất thường có thể là dấu hiệu của sự cố kỹ thuật, hỏng hóc cơ khí hoặc hao mòn linh kiện. Điều này không chỉ ảnh hưởng đến tuổi thọ và hiệu suất vận hành của thiết bị mà còn tiềm ẩn nguy cơ mất an toàn và gây thiệt hại kinh tế.

Với sự phát triển của công nghệ Internet of Things (IoT), việc giám sát thiết bị từ xa trở nên dễ dàng hơn bao giờ hết. Bằng cách tích hợp cảm biến và bộ vi điều khiển vào hệ thống giám sát, người dùng có thể theo dõi tình trạng hoạt động của máy móc theo thời gian thực, từ đó phát hiện và xử lý sự cố một cách chủ động. ESP32 là một vi điều khiển mạnh mẽ, tích hợp Wi-Fi và khả năng xử lý tín hiệu tốt, rất phù hợp cho các ứng dụng IoT. Khi kết hợp với cảm biến gia tốc MPU6050, hệ thống có thể theo dõi độ rung trên nhiều trục, phục vụ cho việc giám sát thiết bị trong công nghiệp hoặc dân dụng.

Xuất phát từ nhu cầu thực tiễn đó, em lựa chọn đề tài "Hệ thống giám sát độ rung với ESP32" nhằm thiết kế một hệ thống đơn giản nhưng hiệu quả, sử dụng ESP32 kết hợp MPU6050 để theo dõi độ rung và gửi cảnh báo thông qua nền tảng Blynk. Đây là bước khởi đầu trong việc ứng dụng công nghệ IoT vào công tác bảo vệ và kéo dài tuổi thọ thiết bị, đặc biệt trong các môi trường yêu cầu độ tin cậy cao.

2. Mục tiêu đề tài

Đo và giám sát độ rung bằng cảm biến MPU6050.

- Phân tích dữ liệu và phát hiện rung vượt ngưỡng bằng ESP32.
- Bật LED cảnh báo khi có rung động bất thường.
- Gửi dữ liệu độ rung lên ứng dụng Blynk để theo dõi từ xa.
- Hỗ trợ bảo trì thiết bị chủ động, nâng cao an toàn và hiệu quả vận hành.

3. Phương pháp nghiên cứu

- Tìm hiểu tài liệu về ESP32, MPU6050 và nền tảng IoT.
- Thiết kế sơ đồ mô phỏng trên Wokwi.
- Thiết kế giao diện Blynk

4. Ứng dụng thực tế

Hệ thống có thể được ứng dụng trong:

- Nhà máy công nghiệp: Giám sát độ rung của động cơ, bơm, máy nén khí để phát hiện hỏng hóc sớm.

- Thiết bị dân dụng: Bảo vệ các thiết bị điện như máy giặt, máy hút bụi khi hoạt động bất thường.
- Phòng server hoặc thiết bị nhạy cảm: Phát hiện chấn động gây nguy hại đến phần cứng hoặc dữ liệu.

PHẦN NỘI DUNG

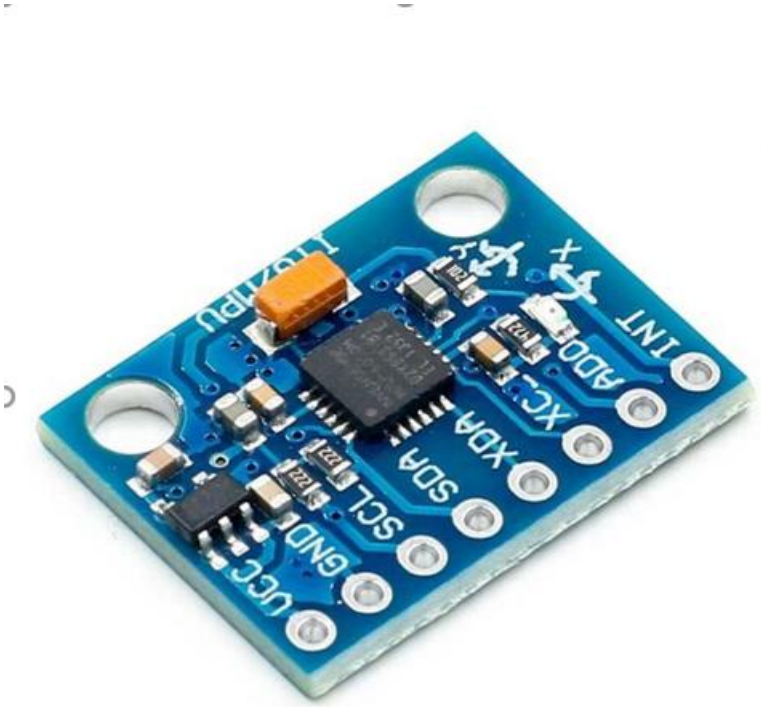
CHƯƠNG 1 : CƠ SỞ LÝ THUYẾT

I.TỔNG QUAN VỀ CẢM BIẾN MPU6050 VÀ ESP32

1.1 MPU6050 – Cảm biến gia tốc và con quay hồi chuyển

1.1.1 MPU6050 là gì ?

MPU6050 là một cảm biến gia tốc kế và con quay hồi chuyển kết hợp trong một module, được sản xuất bởi Invensense. Nó cung cấp các giá trị gia tốc và góc quay của vật thể theo ba trục không gian X, Y và Z. Cảm biến này thường được sử dụng trong các ứng dụng như hệ thống điều khiển cân bằng, máy bay không người lái, và trong các ứng dụng IoT như giám sát rung động.



Cảm biến MPU6050

➢ **Đặc điểm :**

- Tích hợp 2 trong 1 : Bao gồm cảm biến gia tốc 3 trục và con quay hồi chuyển 3 trục
- Giao tiếp I2C : Dễ dàng giao tiếp với vi điều khiển như Arduino,ESP32
- Tích hợp bộ lọc kỹ thuật số : Hỗ trợ xử lý dữ liệu cảm biến ngay trên chip
- Dải đo :
  - Gia tốc kế: ±2g, ±4g, ±8g, ±16g
  - Con quay hồi chuyển: ±250°/s, ±500°/s, ±1000°/s, ±2000°/s
- Điện áp hoạt động : 3.3V hoặc 5V tuỳ module

Thành phần và chức năng từng thành phần của MPU6050

Thành phần	Mô tả và chức năng
Gia tốc kế	Đo gia tốc theo 3 trục (X, Y, Z). Có thể dùng để tính vận tốc, vị trí hoặc phát hiện chuyển động và độ nghiêng.
Con quay hồi chuyển	Đo tốc độ góc (độ/giây) theo 3 trục (X, Y, Z). Dùng để xác định chuyển động quay, định hướng của vật thể.
DMP (Digital Motion Processor)	Bộ xử lý tín hiệu số tích hợp giúp xử lý dữ liệu từ gyro và accelerometer để giảm tải cho vi điều khiển chính. Có thể xử lý lọc Kalman hoặc tích hợp gia tốc + gyro để tính góc nghiêng chính xác hơn.
I2C Interface	Giao tiếp chính với vi điều khiển. Chân SCL (clock) và SDA (data). Địa chỉ I2C mặc định thường là 0x68.
FIFO Buffer	Bộ đệm giúp lưu dữ liệu tạm thời để đọc đồng loạt, tránh mất dữ liệu khi MCU bận xử lý.
Registers	Thanh ghi điều khiển và lưu dữ liệu. Dùng để cấu hình, đọc dữ liệu sensor, đặt ngưỡng, v.v.
Nguồn cung cấp (VCC, GND)	Cung cấp nguồn điện cho module. VCC thường nối 3.3V hoặc 5V, GND nối đất.
Chân INT (Interrupt)	Dùng để tạo ngắt – ví dụ khi có chuyển động, chip có thể gửi tín hiệu cho vi điều khiển biết để xử lý.

1.1.2 Chức năng của gia tốc kế và con quay hồi chuyển

Gia tốc kế (Accelerometer) : Đo độ thay đổi gia tốc của một vật thể trên ba trục X, Y và Z. Nó có thể phát hiện sự di chuyển hoặc thay đổi hướng của vật thể mà nó gắn lên. Gia tốc kế thường được sử dụng để xác định vị trí, độ nghiêng hoặc chuyển động của vật thể.

- **Ứng dụng :** Đo độ nghiêng, đo độ chuyển động của thiết bị , xác định hướng

Con quay hồi chuyển (Gyroscope) : Đo sự thay đổi tốc độ góc quanh ba trục không gian. Con quay hồi chuyển giúp xác định sự quay của thiết bị

- **Ứng dụng :** Đo độ quay hoặc xác định hướng quay, giúp xác định góc quay của các vật thể trong không gian

Sự kết hợp của hai cảm biến này giúp tạo ra hệ thống đo lường chuyển động 6 trục, cực kỳ hữu ích trong việc theo dõi và điều khiển các ứng dụng như robot, xe tự lái, thiết bị đeo, và nhiều ứng dụng khác trong lĩnh vực IoT và thực tế ảo

1.1.3 Ứng dụng thực tế của MPU6050 trong giám sát rung động

MPU6050 có thể được sử dụng trong các ứng dụng giám sát rung động nhờ khả năng đo gia tốc và tốc độ góc của vật thể. Các ứng dụng thực tế bao gồm:

- **Giám sát độ rung của thiết bị:** MPU6050 có thể được gắn lên các máy móc hoặc thiết bị công nghiệp để giám sát các rung động bất thường. Các rung động này có thể là dấu hiệu của sự cố hoặc hỏng hóc sắp xảy ra. Việc phát hiện sớm các rung động có thể giúp giảm thiểu thiệt hại và chi phí sửa chữa.
- **Giám sát tình trạng động cơ và máy móc**: Đo lường và phân tích các rung động trong động cơ hoặc các thiết bị cơ khí giúp dự đoán tình trạng của chúng, giúp bảo trì định kỳ, tránh hư hỏng và giảm thiểu thời gian chết của thiết bị.
- **Ứng dụng trong bảo vệ thiết bị nhạy cảm:** Các thiết bị nhạy cảm như điện thoại di động, máy tính xách tay, thiết bị âm thanh có thể sử dụng MPU6050 để phát hiện các rung động có thể gây hại hoặc làm gián đoạn hoạt động bình thường của thiết bị.
- **Ứng dụng trong đo lường độ ổn định của cấu trúc:** Cảm biến có thể được sử dụng trong các công trình xây dựng hoặc trong giám sát cầu, tòa nhà để phát hiện các rung động bất thường, giúp đảm bảo an toàn trong quá trình sử dụng.

Thông qua các tín hiệu gia tốc và góc quay, MPU6050 có thể giúp theo dõi các rung động theo thời gian thực, giúp các hệ thống tự động nhận diện và xử lý sự cố sớm, tăng cường hiệu quả bảo trì và độ tin cậy của các thiết bị.

1.2 ESP32 – Vi điều khiển dùng trong IoT

1.2.1 Tổng quan về ESP32

ESP32 là một vi điều khiển (MCU) mạnh mẽ và linh hoạt, được sản xuất bởi Espressif Systems. Nó được thiết kế đặc biệt cho các ứng dụng IoT, với khả năng kết nối mạng mạnh mẽ và tiết kiệm năng lượng.

Vi điều khiển ESP32

NodeMCU ESP32 là một bo mạch phát triển dựa trên vi điều khiển **ESP32** của **Espressif Systems**, được thiết kế để dễ dàng lập trình và kết nối với các thiết bị ngoại vi.

- **Đặc điểm chính của NodeMCU ESP32:**

Đặc điểm	Mô tả
Vi điều khiển chính	ESP32 (Dual-core 32-bit, Tensilica LX6)
Tốc độ xử lý	Lên tới 240MHz

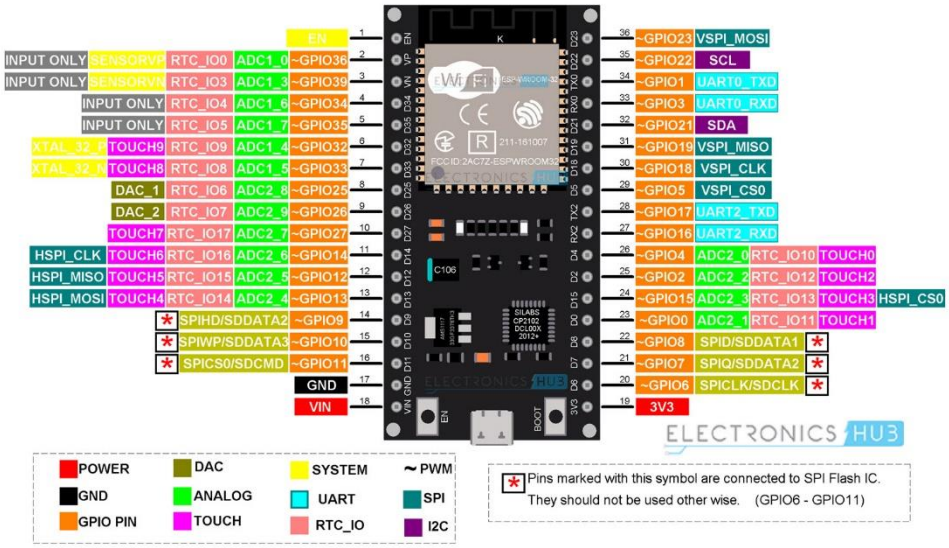
Bộ nhớ	520KB SRAM, 4MB Flash (trên board NodeMCU)
WiFi	Chuẩn 802.11 b/g/n
Bluetooth	Hỗ trợ BLE (Bluetooth Low Energy)
GPIO	Lên đến 34 chân đa năng
ADC/DAC	18 kênh ADC (12-bit), 2 kênh DAC (8-bit)
Giao tiếp hỗ trợ	UART, SPI, I2C, PWM, CAN, IR, v.v.
Nguồn hoạt động	3.3V (USB 5V cấp qua mạch ổn áp
Tích hợp cổng USB	Có thể nạp code dễ dàng qua cổng USB Type B hoặc Micro-USB

Thành phần chính của NodeMCU ESP32

Thành phần	Chức năng
Vi điều khiển ESP32	Bộ xử lý trung tâm, có Wi-Fi & Bluetooth.
Cổng Micro-USB	Kết nối với máy tính để lập trình và cấp nguồn.
Chip USB –TTL (Cp2102/CH340G)	Chuyển đổi tín hiệu từ USB sang UART để giao tiếp với máy tính.
Mạch nguồn (Voltage Regulator)	Chuyển đổi 5V từ USB xuống 3.3V để cấp nguồn cho ESP32.
Nút nhấn (BOOT & RESET)	Dùng để nạp chương trình và khởi động lại ESP32.
Đèn LED tích hợp	LED hiển thị trạng thái hoạt động.
Các chân GPIO (General Purpose Input/Output)	Giao tiếp với cảm biến, module, thiết bị ngoại vi.
Bộ ADC & DAC	Chuyển đổi tín hiệu tương tự sang số và ngược lại
Module Wi-Fi & Bluetooth	Kết nối mạng và giao tiếp không dây.

Chức năng của các chân trên NodeMCU ESP32

ESP32 có nhiều chân GPIO (General Purpose Input/Output) hỗ trợ nhiều chức năng khác nhau



Sơ đồ chân kết nối ESP32

Các chân nguồn và GND

Tên chân	Chức năng
3V3	Cấp nguồn 3.3V cho module/ cảm biến
GND	Nối đất (Ground)
VIN	Cấp nguồn từ ngoài (5V) cho ESP32 hoặc xuất 5V nếu dùng USB

Các chân giao tiếp Digital (GPIO)

ESP32 có 34 chân GPIO, có thể dùng làm Input/Output, PWM, SPI, I2C, UART,...

GPIO	Chức năng chính
GPIO 0	Boot mode, dùng khi nạp firmware.
GPIO 2	Đèn LED tích hợp trên board.
GPIO 5	SPI (CS), có thể dùng làm Output
GPIO 12 , 13, 14,15	SPI (MISO, MOSI, SCK, CS).
GPIO 16,17	Dùng làm UART hoặc Output

⚠ Lưu ý:

- **GPIO 6 → GPIO 11** là các chân kết nối với bộ nhớ Flash, không nên sử dụng.
- **GPIO 34 → GPIO 39** chỉ hỗ trợ Input (không thể làm Output).

Các chân Analog (ADC & DAC)

ESP32 có 18 kênh ADC (độ phân giải 12-bit) và 2 kênh DAC (độ phân giải 8- bit)

Chân	Chức năng
ADC (GPIO 32 – GPIO 39)	Đọc tín hiệu Analog từ cảm biến
DAC (GPIO 25 , GPIO 26)	Xuất tín hiệu Analog

- **Các chân giao tiếp truyền thông**

UART (Giao tiếp Serial)

Chân	Chức năng
GPIO 1 (TX0) , GPIO 3 (RX0)	UART0 – Giao tiếp Serial mặc định
GPIO 9 , GPIO 10	UART1 (ít dùng)
GPIO 16, GPIO 17	UART2

I2C (Giao tiếp với cảm biến, LCD, …)

Chân	Chức năng
GPIO 21	SDA (Data)
GPIO 22	SCL (Clock)

SPI (Giao tiếp với module RF , thẻ nhớ ,…)

Chân	Chức năng
GPIO 23	MOSI (Master Out Slave In)
GPIO 19	MISO (Master In Slave Out)
GPIO 18	SCK (Clock)
GPIO 5	CS (Chip Select)

PWM (Điều chế độ rộng xung , dùng để điều khiển quạt , LED , động cơ servo,…) Tất cả các chân GPIO trên ESP32 đều có thể xuất tín hiệu PWM với tần số và độ phân giải tuỳ chỉnh

1.2.2 Tại sao dùng ESP32 trong IoT?

ESP32 là một sự lựa chọn lý tưởng trong các ứng dụng IoT vì các lý do sau:

- **Tính năng kết nối mạnh mẽ:** ESP32 hỗ trợ cả Wi-Fi và Bluetooth, giúp các thiết bị IoT dễ dàng kết nối với internet và các thiết bị khác. Điều này giúp tạo ra các ứng dụng IoT linh hoạt, từ giám sát từ xa đến điều khiển thiết bị.
- **Tiết kiệm năng lượng:** ESP32 có các chế độ tiết kiệm năng lượng mạnh mẽ, rất thích hợp cho các ứng dụng IoT cần hoạt động lâu dài mà không cần thay đổi pin hoặc sạc lại.
- **Tính linh hoạt và hiệu suất cao:** Với vi xử lý dual-core và khả năng xử lý nhanh, ESP32 có thể xử lý các tác vụ phức tạp như phân tích dữ liệu cảm biến, điều khiển động cơ hoặc giao tiếp với các thiết bị ngoại vi.

- **Chi phí thấp:** ESP32 có giá thành thấp, dễ dàng tiếp cận, phù hợp cho các dự án IoT quy mô nhỏ đến lớn.
- **Phát triển và cộng đồng hỗ trợ mạnh mẽ:** ESP32 được cộng đồng phát triển mạnh mẽ với tài liệu phong phú và nhiều thư viện hỗ trợ, giúp việc lập trình và phát triển ứng dụng dễ dàng hơn.

➢ **<u>Khả năng kết nối WiFi, giao tiếp với MPU6050</u>**

ESP32 có khả năng kết nối Wi-Fi và giao tiếp với cảm biến MPU6050 qua giao thức I2C, cho phép xây dựng các ứng dụng IoT như hệ thống giám sát rung động.

- **Kết nối Wi-Fi:**
  ESP32 có khả năng kết nối với các mạng Wi-Fi, cho phép nó gửi hoặc nhận dữ liệu từ các dịch vụ đám mây hoặc máy chủ từ xa. Ví dụ, ESP32 có thể gửi dữ liệu cảm biến từ MPU6050 đến một ứng dụng web hoặc dịch vụ như ThingSpeak hoặc Blynk để giám sát và phân tích từ xa.
- **Giao tiếp với MPU6050:**
  ESP32 sử dụng giao tiếp I2C để kết nối với cảm biến MPU6050, truyền tải dữ liệu gia tốc và góc quay từ cảm biến này tới vi điều khiển. ESP32 có các chân I2C (SDA, SCL) có thể dễ dàng kết nối với MPU6050, giúp thu thập dữ liệu cảm biến và xử lý hoặc gửi dữ liệu này đến các nền tảng đám mây.
- **Lợi ích:** Việc sử dụng I2C giúp đơn giản hóa việc kết nối và truyền tải dữ liệu, đồng thời tiết kiệm số lượng chân kết nối trên ESP32.

1.2.3 Ứng dụng ESP32 trong giám sát rung động

ESP32 rất hữu ích trong việc xây dựng các hệ thống giám sát rung động nhờ khả năng kết nối mạng và xử lý tín hiệu mạnh mẽ. Dưới đây là các ứng dụng thực tế:

- **Giám sát độ rung thiết bị công nghiệp:** ESP32 có thể được gắn lên các thiết bị hoặc máy móc trong nhà máy để theo dõi mức độ rung động. Cảm biến MPU6050 sẽ đo các rung động này và ESP32 có thể gửi dữ liệu này đến một ứng dụng trên đám mây (như ThingSpeak hoặc Blynk) để phân tích và cảnh báo người quản lý nếu có sự bất thường.
- **Giám sát môi trường:** ESP32 có thể giám sát rung động trong môi trường xây dựng, chẳng hạn như các tòa nhà hoặc cầu đường. Việc theo dõi độ rung có thể giúp phát hiện các dấu hiệu của sự cố hoặc nguy cơ hư hỏng.
- **Ứng dụng trong các thiết bị đeo:** Trong các thiết bị đeo, như đồng hồ thông minh hoặc vòng tay theo dõi sức khỏe, ESP32 có thể theo dõi các rung động

và gửi dữ liệu về tình trạng sức khỏe người dùng (như theo dõi rung động tay khi đi bộ hoặc chạy).

- **Giám sát và bảo vệ thiết bị điện tử nhạy cảm:** ESP32 có thể phát hiện các rung động bất thường có thể gây hư hỏng cho các thiết bị điện tử nhạy cảm như điện thoại di động, máy tính xách tay, hoặc các thiết bị trong phòng thí nghiệm.
- **Điều khiển và bảo trì từ xa:** Với khả năng kết nối Wi-Fi, ESP32 có thể tạo ra hệ thống giám sát rung động từ xa, giúp các kỹ sư theo dõi tình trạng của thiết bị mà không cần phải có mặt trực tiếp tại hiện trường.

1.3 Tổng quan về IoT và ứng dụng trong giám sát rung động

1.3.1 IoT là gì?

IoT (Internet of Things), hay "Internet vạn vật," là một mạng lưới các thiết bị vật lý có thể kết nối và giao tiếp với nhau qua internet hoặc các mạng truyền thông khác mà không cần sự can thiệp của con người. Các thiết bị này có thể là bất kỳ vật dụng nào, từ điện thoại thông minh, máy tính, cảm biến, thiết bị gia đình, đến các máy móc công nghiệp.

➢ **Đặc điểm của IoT:**

- **Kết nối mạng:** Các thiết bị IoT có thể kết nối qua các giao thức mạng như Wi-Fi, Bluetooth, Zigbee, hoặc LoRa, cho phép chúng truyền tải và nhận dữ liệu từ các máy chủ hoặc các nền tảng đám mây.
- **Thu thập và phân tích dữ liệu:** Các thiết bị IoT có thể thu thập dữ liệu từ môi trường hoặc các cảm biến, sau đó gửi dữ liệu này đến các máy chủ để phân tích và ra quyết định tự động hoặc cảnh báo.
- **Tự động hóa:** IoT giúp tự động hóa các tác vụ, giúp cải thiện hiệu suất, giảm thiểu sai sót và tăng cường sự thuận tiện cho người dùng.
- **Ứng dụng rộng rãi:** IoT được ứng dụng trong nhiều lĩnh vực, bao gồm nhà thông minh, giao thông thông minh, chăm sóc sức khỏe, sản xuất và công nghiệp.

1.3.2 Các phương pháp giám sát rung động trong công nghiệp

Giám sát rung động trong công nghiệp là một phần quan trọng trong bảo trì và vận hành máy móc, giúp phát hiện các sự cố và giảm thiểu thời gian ngừng hoạt động. Các phương pháp giám sát rung động phổ biến bao gồm:

- ***Giám sát rung động bằng cảm biến gia tốc (Accelerometer):***
  - **Phương pháp:** Sử dụng cảm biến gia tốc để đo mức độ rung động của máy móc và thiết bị. Cảm biến này có thể gắn vào các bộ phận của máy móc để phát hiện sự thay đổi bất thường trong chuyển động hoặc độ rung.

- **Ưu điểm:** Cảm biến gia tốc nhỏ gọn và dễ triển khai. Chúng có thể phát hiện sự thay đổi về gia tốc và các rung động do hư hỏng hoặc sự bất thường trong hệ thống.

- ***Giám sát rung động bằng cảm biến gia tốc và con quay hồi chuyển (IMU - Inertial Measurement Unit):***
  - **Phương pháp:** Kết hợp cảm biến gia tốc và con quay hồi chuyển để theo dõi không chỉ độ rung mà còn cả sự quay hoặc thay đổi góc của thiết bị.
  - **Ứng dụng:** Thường được sử dụng trong các hệ thống giám sát động cơ, máy bơm, máy nén khí, và các thiết bị công nghiệp khác.
- ***Phân tích dải tần số (Frequency Spectrum Analysis):***
  - **Phương pháp:** Sử dụng phân tích Fourier để phân tích dải tần số của rung động và xác định nguồn gốc của các rung động (như lỗi vòng bi, ròng rọc hoặc các vấn đề cơ khí khác).
  - **Ứng dụng:** Dùng để phát hiện các rung động không bình thường, phân tích mức độ hư hỏng của các bộ phận máy móc như động cơ và máy bơm.
- ***Giám sát bằng cảm biến từ trường (Magnetic Sensors):***
  - **Phương pháp:** Cảm biến từ trường có thể đo sự thay đổi của trường từ do các bộ phận của máy móc gây ra khi bị rung động hoặc xáo trộn.
  - **Ứng dụng:** Thường được sử dụng trong các ứng dụng giám sát động cơ điện hoặc các hệ thống có từ trường mạnh.
- ***Phân tích âm thanh và sóng siêu âm:***
  - **Phương pháp:** Sử dụng micrô và cảm biến âm thanh để phát hiện các âm thanh bất thường, có thể là dấu hiệu của các vấn đề cơ học như sự mài mòn hoặc lệch tâm.
  - **Ứng dụng:** Giám sát các máy bơm, quạt, hoặc các thiết bị có chuyển động quay.

1.3.3 Ứng dụng của IoT trong bảo trì thiết bị

IoT giúp cải thiện quy trình bảo trì thiết bị trong công nghiệp nhờ khả năng giám sát, thu thập và phân tích dữ liệu thời gian thực. Dưới đây là các ứng dụng cụ thể:

- ***Bảo trì dựa trên tình trạng (Condition-based Maintenance):***
  - **Phương pháp:** Sử dụng cảm biến IoT để giám sát các thông số quan trọng của thiết bị, như nhiệt độ, áp suất, độ rung, và các yếu tố khác. Dữ liệu từ các cảm biến này được gửi đến các hệ thống phân tích để xác định tình trạng hoạt động của thiết bị.

- **Ứng dụng:** Giúp xác định khi nào thiết bị cần bảo trì, thay vì bảo trì theo định kỳ. Điều này giúp giảm thiểu chi phí bảo trì và thời gian ngừng hoạt động.

- ***Bảo trì dự đoán (Predictive Maintenance):***
  - **Phương pháp:** Dựa trên dữ liệu thu thập được từ các cảm biến IoT, các thuật toán phân tích và học máy có thể dự đoán thời gian còn lại của thiết bị trước khi hỏng hóc. Điều này giúp lên kế hoạch bảo trì trước khi sự cố xảy ra.
  - **Ứng dụng:** Giúp giảm thiểu rủi ro hỏng hóc và tránh được các sự cố khẩn cấp, đồng thời kéo dài tuổi thọ của thiết bị.
- ***Giám sát từ xa:***
  - **Phương pháp:** Các hệ thống IoT có thể giám sát và điều khiển thiết bị từ xa qua các nền tảng đám mây. Điều này giúp các kỹ sư và nhà vận hành theo dõi tình trạng thiết bị mà không cần phải có mặt tại hiện trường.
  - **Ứng dụng:** Các hệ thống giám sát IoT có thể cảnh báo khi thiết bị gặp sự cố hoặc yêu cầu bảo trì, giúp tối ưu hóa thời gian và chi phí.
- ***Tối ưu hóa hiệu suất thiết bị:***
  - **Phương pháp:** Dữ liệu IoT có thể được sử dụng để tối ưu hóa hiệu suất của thiết bị thông qua việc điều chỉnh các tham số hoạt động như tốc độ, áp suất, hoặc nhiệt độ dựa trên các dữ liệu thu thập được.
  - **Ứng dụng:** Giúp tiết kiệm năng lượng và giảm thiểu hao mòn thiết bị.
- ***Theo dõi tình trạng thiết bị trong thời gian thực:***
  - **Phương pháp:** Các thiết bị IoT có thể thu thập dữ liệu liên tục và gửi về các trung tâm điều khiển để phân tích. Dựa vào các chỉ số này, các nhà vận hành có thể can thiệp sớm nếu phát hiện ra vấn đề.
  - **Ứng dụng:** Các máy móc như máy nén khí, máy bơm, động cơ có thể được theo dõi và bảo trì hiệu quả hơn.

II . TỔNG QUAN VỀ PlatformIO,WOKWI VÀ BLYNK

2.1 Giới thiệu về PlatformIO

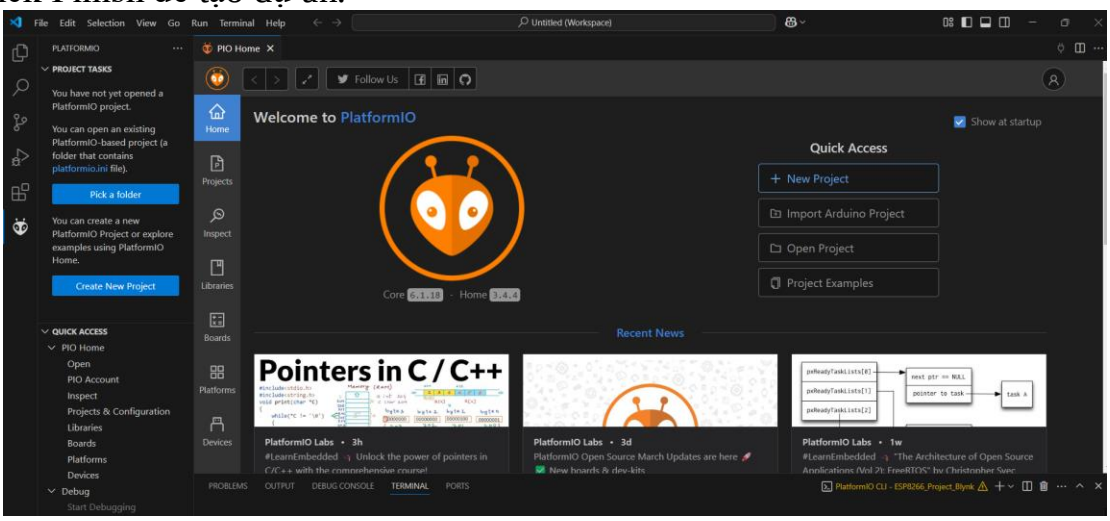
PlatformIO là một môi trường phát triển tích hợp (IDE) mã nguồn mở, dùng để lập trình vi điều khiển như ESP32, Arduino, STM32, ESP8266 và nhiều loại vi điều khiển khác. PlatformIO có thể chạy độc lập hoặc tích hợp vào các trình soạn thảo mã nguồn như Visual Studio Code (VS Code).

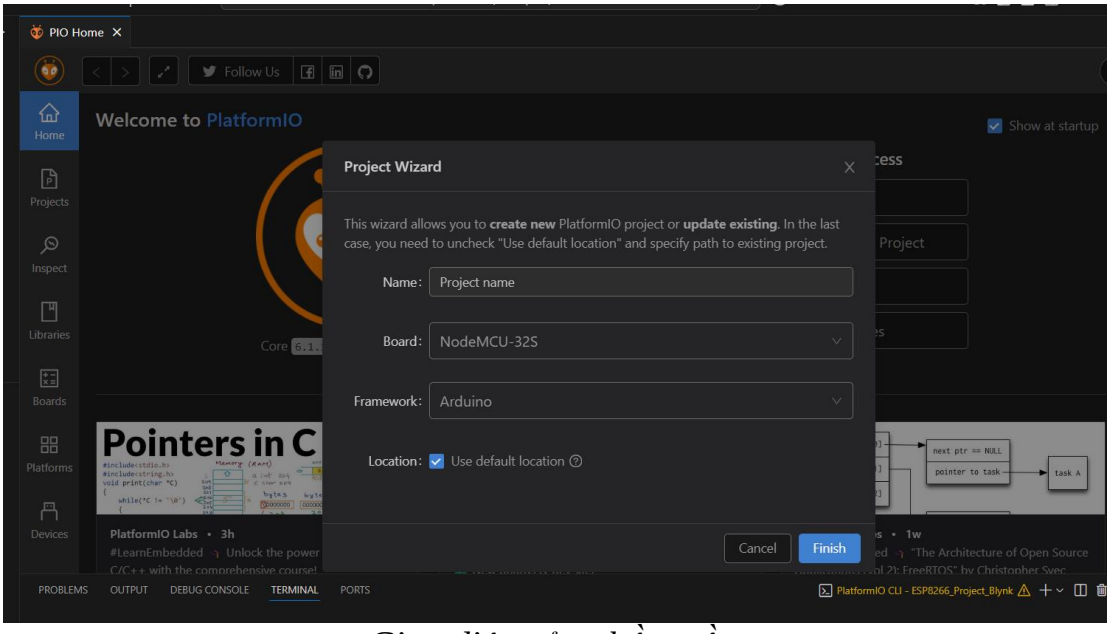
Đặc điểm chính của PlatformIO:

- Đa nền tảng: Hỗ trợ hơn 40 nền tảng phát triển khác nhau, bao gồm Arduino, ESP32, STM32, và nhiều hơn nữa, với hơn 20 framework như Arduino, ESPIDF, và CMSIS.
- Quản lý thư viện thông minh: Hệ thống quản lý thư viện tự động tìm kiếm, cài đặt và cập nhật các thư viện cần thiết cho dự án.
- Tích hợp với nhiều IDE: PlatformIO có thể tích hợp với nhiều IDE phổ biến như Visual Studio Code, Atom, CLion, và Eclipse, mang lại trải nghiệm phát triển linh hoạt.
- Hệ thống xây dựng thống nhất: Cung cấp hệ thống xây dựng thống nhất cho tất cả các nền tảng, đơn giản hóa quy trình phát triển đa nền tảng.
- Công cụ debug nâng cao: Tích hợp các công cụ debug mạnh mẽ, bao gồm debug phần cứng thông qua các giao thức như JTAG và SWD.
- Kiểm thử tự động: Hỗ trợ kiểm thử đơn vị và tích hợp liên tục (CI) cho các dự án nhúng.
- Quản lý phiên bản: Tích hợp với các hệ thống quản lý phiên bản như Git, giúp theo dõi và quản lý các thay đổi trong mã nguồn.

Cách thực hiện :

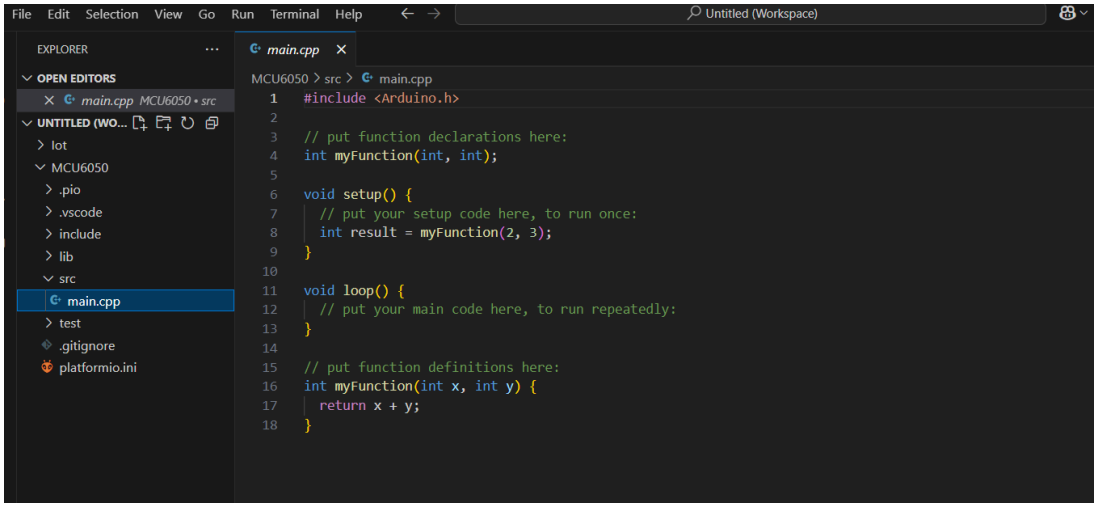
+ Mở VSCode → Click vào biểu tượng PlatformIO.
+ Chọn New Project.
+ Đặt tên dự án, chọn Board: ESP32, Framework: Arduino.
+ Click Finish để tạo dự án.





Giao diện của phần mềm

+ Tiếp theo vào src/main.cpp và nhập code



2.2 Công cụ mô phỏng Wokwi

2.2.1 Wokwi là gì ?

Wokwi là một nền tảng mô phỏng mạch điện tử và lập trình vi điều khiển trực tuyến, cho phép người dùng thử nghiệm và phát triển các dự án IoT (Internet of Things) mà không cần phần cứng thực tế. Wokwi hỗ trợ các loại vi điều khiển phổ biến như Arduino Uno, ESP32, ESP8266, cũng như nhiều loại cảm biến và linh kiện điện tử khác như LED, cảm biến nhiệt độ, cảm biến siêu âm, màn hình LCD, module Wi-Fi, v.v..

2.2.2 Ưu điểm của Wokwi trong mô phỏng hệ thống IoT

Ưu điểm	Mô tả
Dễ sử dụng	Giao diện trực quan, dễ kéo thả linh kiện, viết code và chạy mô phỏng ngay trên trình duyệt.
Không cần phần cứng thật	Giúp sinh viên và người học có thể thử nghiệm các mạch điện tử mà không cần mua thiết bị thật, tiết kiệm chi phí.
Hỗ trợ nhiều vi điều khiển phổ biến	Wokwi hỗ trợ các loại chip phổ biến trong IoT như Arduino, ESP32, Raspberry Pi Pico… giúp mô phỏng sát với thực tế triển khai.
Mô phỏng thời gian thực	Các giá trị cảm biến, trạng thái chân GPIO, xung PWM… được cập nhật và hiển thị theo thời gian thực.
Hỗ trợ lập trình với Arduino IDE và PlatformIO	Người dùng có thể viết code theo chuẩn Arduino/C++ và mô phỏng trực tiếp. Ngoài ra, có thể tải file project để chạy bằng PlatformIO.
Tích hợp dễ dàng với các dịch vụ IoT	Người dùng có thể kết nối mô phỏng với các dịch vụ như Blynk, ThingSpeak, MQTT, từ đó kiểm tra giao tiếp mạng của hệ thống IoT.
Cộng đồng hỗ trợ mạnh	Wokwi có diễn đàn và cộng đồng người dùng sôi động, nhiều ví dụ mẫu và hướng dẫn có sẵn.

2.3 Blynk

2.3.1 Blynk là gì ?

Blynk là một nền tảng IoT (Internet of Things) mạnh mẽ và linh hoạt, cho phép người dùng dễ dàng xây dựng các ứng dụng điều khiển và giám sát thiết bị phần cứng từ xa thông qua điện thoại thông minh hoặc máy tính bảng.

Blynk cung cấp một ứng dụng di động (iOS/Android) và một nền tảng điện toán đám mây, hỗ trợ kết nối với các vi điều khiển như ESP32, ESP8266, Arduino, Raspberry Pi,... thông qua mạng Wi-Fi hoặc Internet. Người dùng có thể kéo-thả các widget (nút nhấn, đồ thị, hiển thị số, gauge...) để tạo giao diện điều khiển mà không cần viết ứng dụng từ đầu.

2.3.2 Ưu điểm của Blynk trong IoT

Ưu điểm	Mô tả
Dễ sử dụng và giao diện thân thiện	Không cần kỹ năng lập trình ứng dụng mobile. Giao diện kéo-thả giúp thiết kế dashboard trực quan chỉ trong vài phút.
Hỗ trợ đa nền tảng phần cứng	Làm việc tốt với nhiều loại vi điều khiển, đặc biệt là ESP32, rất phổ biến trong các dự án IoT.
Kết nối thời gian thực	Giao tiếp hai chiều giữa thiết bị và ứng dụng: có thể giám sát dữ liệu cảm biến và điều khiển thiết bị từ xa.
Tích hợp với Blynk Cloud hoặc server riêng	Dữ liệu được lưu trữ và truyền tải an toàn qua Blynk Cloud, hoặc có thể tùy chọn cài đặt server riêng để chủ động hơn.
Tích hợp biểu đồ, ghi log dữ liệu , thông báo cảnh báo	Giúp theo dõi sự thay đổi của cảm biến theo thời gian, phát hiện nhanh các tình huống bất thường.
Hỗ trợ mã nguồn mở và cộng đồng lớn	Có tài liệu rõ ràng và ví dụ mẫu phong phú, dễ học và triển khai thực tế.

2.3.3 Lợi ích khi sử dụng Blynk trong giám sát rung động

Lợi ích	Mô tả
Giám sát từ xa qua điện thoại	Người dùng có thể theo dõi mức độ rung động của thiết bị mọi lúc, mọi nơi chỉ với một chiếc smartphone có kết nối mạng.
Cảnh báo thời gian thực	Có thể cấu hình để gửi thông báo khi rung vượt ngưỡng cho phép, giúp phát hiện hỏng hóc hoặc bất thường kịp thời.
Hiển thị trực quan	Sử dụng các widget như biểu đồ (Chart), gauge, giá trị số, giúp hình dung được mức độ rung động theo thời gian.
Lưu trữ và phân tích dữ liệu	Có thể ghi log dữ liệu rung động, hỗ trợ việc phân tích hành vi rung để bảo trì thiết bị hiệu quả hơn.

Tăng tính cơ động và tiết kiệm chi phí	Không cần máy tính chuyên dụng hay phần mềm phức tạp, chỉ cần ESP32 + cảm biến + ứng dụng Blynk là đủ.

CHƯƠNG 2 : NGUYÊN LÝ HOẠT ĐỘNG

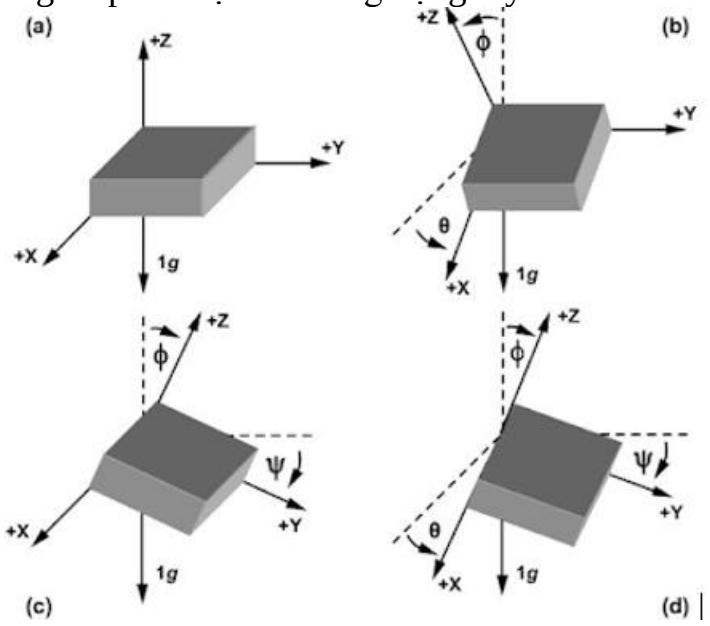
I. CÁCH MPU6050 ĐO RUNG ĐỘNG

1.1 Nguyên lý hoạt động của gia tốc kế

Gia tốc kế đo gia tốc tuyến tính (linear acceleration) của vật thể theo ba trục X, Y và Z. Nó hoạt động dựa trên nguyên lý:

- Khi vật thể bị rung hoặc chuyển động, sẽ tạo ra một lực tác động lên các khối lượng nhỏ (mass) bên trong gia tốc kế.
- Sự dịch chuyển của các khối lượng này sẽ làm thay đổi điện dung (capacitive) hoặc điện trở, từ đó chuyển đổi thành tín hiệu điện.

➞ MPU6050 sử dụng gia tốc kế MEMS (Micro-Electro-Mechanical System) để đo gia tốc theo 3 hướng và phát hiện các rung động tuyến tính.



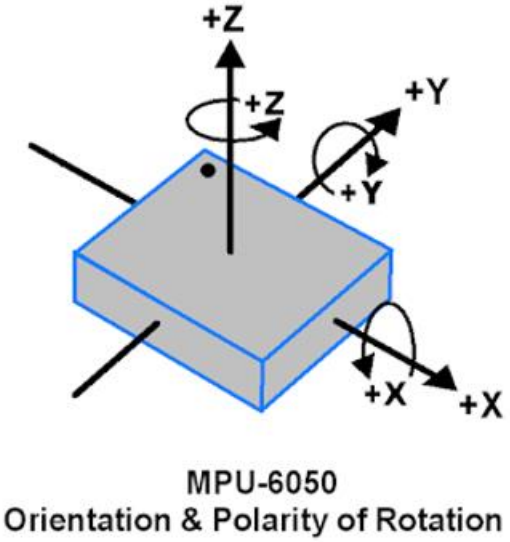
Hình ảnh minh hoạ

1.2 Nguyên lý hoạt động của con quay hồi chuyển

Con quay hồi chuyển đo tốc độ góc (angular velocity) - tức là mức độ xoay của vật thể quanh các trục X, Y, Z.

Khi có sự xoay chuyển, phần tử MEMS bên trong cảm biến sẽ tạo ra lực Coriolis làm thay đổi vị trí hoặc điện áp, từ đó xác định được tốc độ quay.

➞ Điều này giúp MPU6050 nhận biết chuyển động quay hoặc rung động kiểu xoay tròn, thường gặp khi thiết bị rung lắc trong môi trường công nghiệp hoặc cơ khí.



MPU-6050
Orientation & Polarity of Rotation

Hình ảnh minh hoạ

1.3 Công thức tính gia tốc tổng hợp

Khi đo rung động tổng thể, người ta thường sử dụng **gia tốc tổng hợp (total acceleration)**, được tính bằng công thức:

$$a_{total} = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Trong đó:

- a_x, a_y, a_z là các giá trị gia tốc đo được theo ba trục (đơn vị: m/s² hoặc g).
- a_{total}thể hiện mức độ rung động tổng thể của thiết bị.

Nếu giá trị a_{total} dao động mạnh hoặc vượt ngưỡng ngưỡng đặt trước, có thể xem là thiết bị đang bị **rung mạnh bất thường**.

1.4 Ứng dụng trong bảo vệ thiết bị

MPU6050 có thể được ứng dụng để giám sát độ rung của các thiết bị máy móc, động cơ, tủ điện hoặc robot. Cụ thể:

- Phát hiện rung động bất thường: Cảnh báo sớm nếu thiết bị rung mạnh bất thường do lỏng ốc, mất cân bằng hoặc hư hỏng.
- Giảm thiểu thiệt hại: Kết hợp với ESP32 và Blynk, hệ thống có thể gửi cảnh báo ngay lập tức đến người quản lý để kiểm tra thiết bị.

- Theo dõi liên tục và bảo trì dự đoán: Lưu lại dữ liệu rung để phân tích xu hướng – khi rung tăng dần theo thời gian có thể là dấu hiệu mài mòn, cần bảo trì sớm.

II. TỔNG QUAN HỆ THỐNG

2.1 Tổng quan hệ thống

Hệ thống này sử dụng vi điều khiển ESP32 để đọc dữ liệu rung động từ cảm biến gia tốc MPU6050. Khi độ rung vượt quá một ngưỡng nhất định (ví dụ: biên độ gia tốc > 2g), ESP32 sẽ ghi nhận tình trạng bất thường và gửi dữ liệu này lên ứng dụng Blynk thông qua WiFi.

Người dùng có thể theo dõi mức độ rung của thiết bị theo thời gian thực thông qua giao diện trực quan trên điện thoại hoặc trình duyệt. Việc này giúp phát hiện sớm các hư hỏng, trục trặc có thể xảy ra đối với thiết bị được giám sát.

2.2 Sơ đồ khối hệ thống

Hệ thống bao gồm các thành phần chính sau:

- **Cảm biến gia tốc MPU6050:** Đo gia tốc (rung động) của thiết bị và gửi dữ liệu về ESP32 thông qua giao tiếp I2C.
- **ESP32:** Vi điều khiển trung tâm, nhận và xử lý dữ liệu từ MPU6050. So sánh giá trị rung với ngưỡng an toàn để quyết định kích hoạt cảnh báo. Đồng thời gửi dữ liệu lên Blynk qua Wi-Fi.
- **LED cảnh báo:** Được điều khiển bởi ESP32, dùng để báo hiệu khi mức độ rung vượt ngưỡng cho phép (LED sáng khi có rung mạnh).

2.3 Nguyên lý hoạt động chi tiết

2.3.1 Đọc dữ liệu từ cảm biến gia tốc MPU6050

- Cảm biến MPU6050 có khả năng đo gia tốc 3 trục và con quay hồi chuyển (gyro)
- Trong hệ thống này, cảm biến được sử dụng để phát hiện độ rung hoặc dao động của thiết bị
- ESP32 kết nối với MPU6050 qua giao tiếp I2C (chân SDA và SCL)
- Tại mỗi chu kì đọc, ESP32 lấy giá trị gia tốc từ cảm biến

2.3.2 Xử lý dữ liệu trên ESP32

- ESP32 xử lý giá trị gia tốc nhận được từ MPU6050 để xác định mức độ rung

- Giá trị rung được quy đổi ra chỉ số tổng hợp (tạm gọi là " gia tốc rung") để dễ so sánh
- Nếu gia tốc rung vượt ngưỡng nhất định (ví dụ : 50) , ESP32 kích hoạt cảnh báo
- Nếu rung giảm xuống dưới mức an toàn (ví dụ : < 30), cảnh báo sẽ được tắt
- Việc sử dụng ngưỡng trễ (hysteresis) giúp tránh việc cảnh báo nhấp nhánh liên tục khi giá trị rung giao động gần ngưỡng

2.3.3 Điều khiển LED cảnh báo

- ESP32 điều khiển một đèn LED thông qua tín hiệu digital output
- Khi rung vượt ngưỡng => LED sáng để cảnh báo
- Khi rung ổn định dưới ngưỡng => LED tắt

2.3.4 Gửi dữ liệu lên Blynk để giám sát từ xa

- ESP32 kết nối Wi-Fi và gửi dữ liệu lên nền tảng Blynk Cloud.
- Trên ứng dụng Blynk:
  - Widget "Gia tốc rung" (V2) hiển thị mức rung hiện tại theo thời gian thực.
  - Widget "Alert" (V1) hiển thị cảnh báo (biểu tượng đèn đỏ khi rung mạnh).
- Người dùng có thể giám sát trạng thái rung từ xa thông qua điện thoại hoặc trình duyệt.

2.4 Mô tả hoạt động theo sơ đồ trạng thái

Trạng thái	Điều kiện	Hành động của ESP32	Trạng thái LED cảnh báo
Chờ dữ liệu cảm biến	ESP32 đọc dữ liệu từ MPU6050	Lưu giá trị gia tốc rung vào biến	-
Rung yếu	Giá trị rung < 30	Tắt LED cảnh báo	LED OFF
Rung mạnh	Giá trị rung ≥ 50	Bật LED cảnh báo	LED ON
Rung trung bình (ngưỡng trễ)	30 ≤ Giá trị rung < 50	Giữ trạng thái LED trước đó	Không thay đổi

CHƯƠNG 3 : THIẾT KẾ HỆ THỐNG

I.SƠ ĐỒ KẾT NỐI MÔ PHỎNG

1.1Thành phần chính

Hệ thống bao gồm các linh kiện chính sau :

- **ESP32:** Vi điều khiển chính, có khả năng kết nối Wi-Fi, thu thập và xử lý dữ liệu từ cảm biến MPU6050, đồng thời điều khiển LED cảnh báo.
- **MPU6050:** Cảm biến đo gia tốc 3 trục và con quay hồi chuyển, dùng để phát hiện rung động của thiết bị.
- **LED:** Dùng để hiển thị trạng thái cảnh báo khi phát hiện rung động vượt mức ngưỡng cho phép

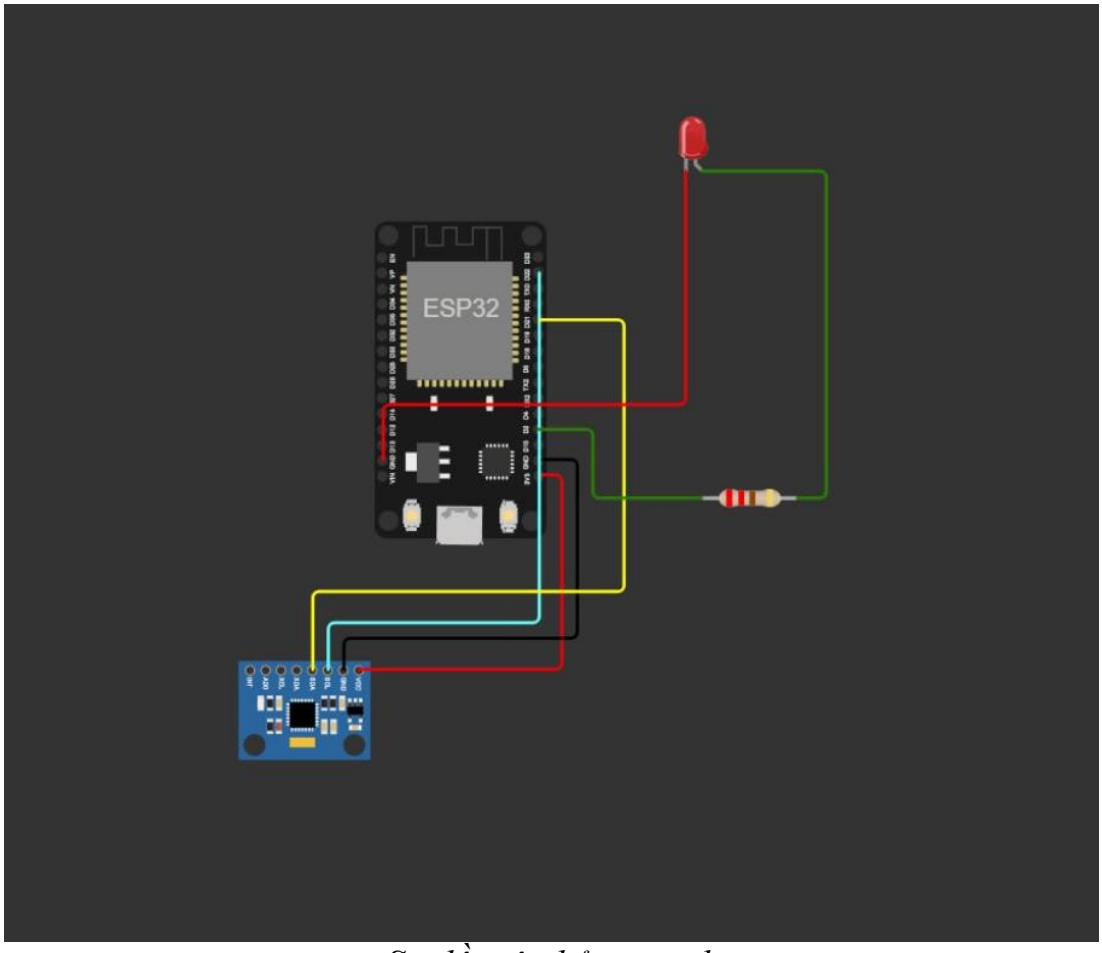
1.2 Sơ đồ mạch

MPU6050:
+ VCC → 3.3V ESP32
+ GND → GND ESP32
+ SDA → GPIO21 ESP32
+ SCL → GPIO22 ESP32

LED cảnh báo :
+ Anode (chân dương) → GPIO15 ESP32 (qua điện trở ~220Ω)
+ Cathode (chân âm) → GND ESP32

Trong môi trường mô phỏng Wokwi, các linh kiện này được định nghĩa trong tệp cấu hình JSON như sau :

```
{
  "version": 1,
  "author": "Nguyễn Bình Khánh Nhiên ",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp32", "top": 0, "left":
0, "attrs": {} },
    { "type": "wokwi-mpu6050", "id": "mpu", "top": 272.62, "left":
-84.08, "attrs": {} },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": -70.8,
      "left": 176.6,
      "attrs": { "color": "red" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": 167.15,
      "left": 201.6,
      "attrs": { "value": "220" }
    }
  ],
  "connections": [
    [ "esp32:3V3", "mpu:VCC", "red", [ "h13.9", "v119.9" ] ],
    [ "esp32:GND.1", "mpu:GND", "black", [ "h23.5", "v110.2", "h-144.08"
 ] ],
    [ "esp32:D22", "mpu:SCL", "cyan", [ "v215.6", "h-130.18" ] ],
    [ "esp32:D21", "mpu:SDA", "yellow", [ "h52.3", "v167.5", "h-192.08"
 ] ],
    [ "esp32:D2", "r1:1", "green", [ "h33.1", "v42.4", "h67.2" ] ],
    [ "r1:2", "led1:A", "green", [ "h18", "v-201.6" ] ],
    [ "led1:C", "esp32:GND.2", "red", [ "v144", "h-186.6" ] ]
  ],
  "dependencies": {}
}
```



Sơ đồ mô phỏng mạch

Chi tiết kết nối:

Các thành phần trong hệ thống được kết nối với nhau theo sơ đồ mạch đã được thiết kế trên nền tảng Wokwi. Các kết nối này được định nghĩa trong phần "Connections" của tệp cấu hình JSON :

```
  "connections": [
    [ "esp32:3V3", "mpu:VCC", "red", [ "h13.9", "v119.9" ] ],
    [ "esp32:GND.1", "mpu:GND", "black", [ "h23.5", "v110.2", "h-144.08"
] ],
    [ "esp32:D22", "mpu:SCL", "cyan", [ "v215.6", "h-130.18" ] ],
    [ "esp32:D21", "mpu:SDA", "yellow", [ "h52.3", "v167.5", "h-192.08"
] ],
    [ "esp32:D2", "r1:1", "green", [ "h33.1", "v42.4", "h67.2" ] ],
    [ "r1:2", "led1:A", "green", [ "h18", "v-201.6" ] ],
    [ "led1:C", "esp32:GND.2", "red", [ "v144", "h-186.6" ] ]
  ],
  "dependencies": {}
}
```

- ESP32 cấp nguồn 3.3V cho cảm biến MPU6050, giúp cảm biến hoạt động ổn định
- Chân SDA và SCL của MPU6050 kết nối với GPIO21 và GPIO22 của ESP32, truyền dữ liệu qua giao tiếp I2C
- Chân dương của LED nối với GPIO15, cho phép ESP32 điều khiển bật / tắt LED cảnh báo khi rung vượt ngưỡng
- Chân âm của LED nối về GND, hoàn tất mạch điều khiển LED
- Nguồn LED được cấp từ ESP32, an toàn và phù hợp với mức điện áp của hệ thống

II.THIẾT KẾ GIAO DIỆN BLYNK

2.1 Cấu hình Blynk

2.2.1 Tạo tài khoản và dự án trên Blynk

+ Tạo tài khoản

Đăng nhập Blynk web hoặc tải ứng dụng Blynk trên điện thoại

Tạo tài khoản bằng email của bạn

+ Tạo dự án mới

Mở ứng dụng Blynk, đăng nhập và nhấn vào **" Template**" => + New Template

Chọn Hardware ESP32

Connection type : Wifi

Name : ESP32 MPU6050

Nhấn Done

+ Tạo Datastreams

Chọn Template vừa tạo => **Datastreams** => + New Datastream:

- **Gia tốc rung**

Type : Virtual Pin

Pin : V2

Data Type : Integer

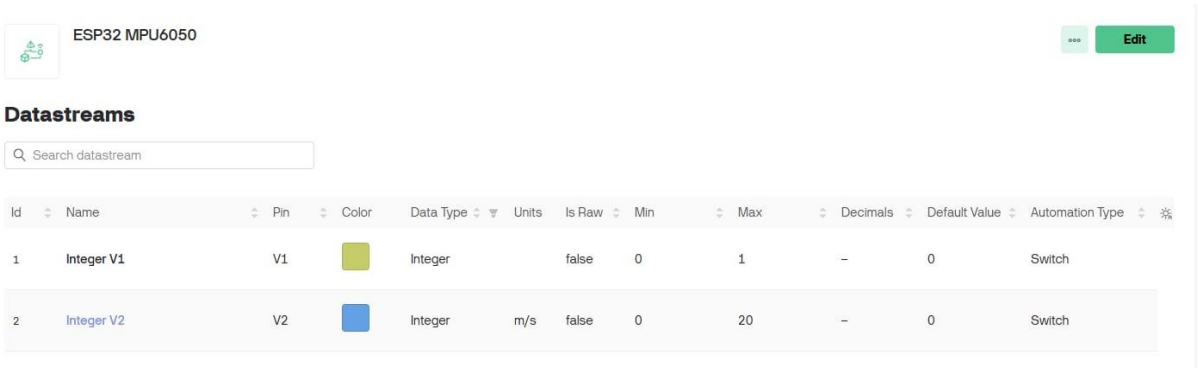
Min : 0 Max: 20

- **Trạng thái cảnh báo :**

Type : Virtual Pin

Pin: V1

Data Type : Integer



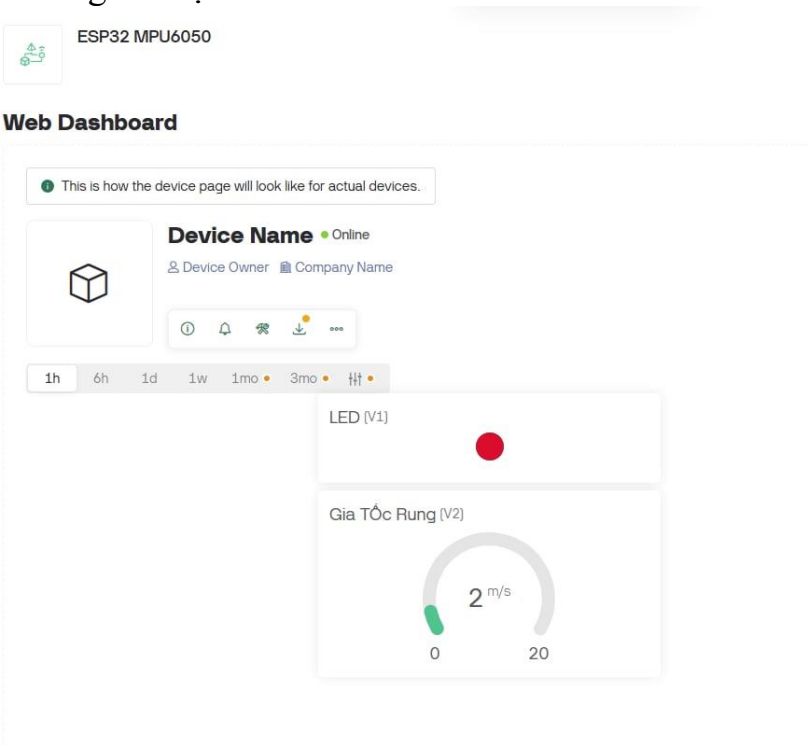
Datastreams đã được tạo

+ Tạo giao diện điều khiển (Web Dashboard)

- Trong ứng dụng Blynk, vào phần " Web Dashboard"
- ***Thêm Gauge :***
  - Kéo thả một Gauge, gắn nó với Datastream V2
  - Đặt tên : " Gia tốc rung"

- ***Thêm LED***
  - Kéo thả một LED, gắn nó với Datastream V1

Nhấn **"Save"** để lưu giao diện



Giao diện Dashboard sau khi được tạo

III. MÃ LẬP TRÌNH TRÊN VSCODE

3.1 Tổng quan

Mã code ở file src/main.cpp trong dự án PlatformIO:

```cpp
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

/* Thông tin Blynk */
#define BLYNK_TEMPLATE_ID "TMPL6Bmf8f_0b"
#define BLYNK_TEMPLATE_NAME "ESP32 MPU6050"
#define BLYNK_AUTH_TOKEN "uTYeUxybdAzm0-VZ6EgH1cj9Gc6tPPQv"

// Thông tin WiFi cho Wokwi
char ssid[] = "Wokwi-GUEST";
char pass[] = "";

// Định nghĩa chân
#define LED_PIN 2  // Chân kết nối LED

// Khởi tạo MPU6050
Adafruit_MPU6050 mpu;

// Biến toàn cục
float threshold = 1.5;  // Ngưỡng rung động (m/s²)
bool vibrationDetected = false;  // Trạng thái rung động
unsigned long currentMiliseconds = 0;

// Hàm kiểm tra thời gian
bool IsReady(unsigned long &ulTimer, uint32_t milisecond) {
  if (currentMiliseconds - ulTimer < milisecond) return false;
  ulTimer = currentMiliseconds;
  return true;
}

// Cập nhật trạng thái rung động
void updateVibration() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  // Tính độ lớn gia tốc tổng
  float accel = sqrt(a.acceleration.x * a.acceleration.x +
                     a.acceleration.y * a.acceleration.y +
                     a.acceleration.z * a.acceleration.z);

  // Kiểm tra rung động
  if (accel > threshold) {
    if (!vibrationDetected) {
      Serial.println("Phát hiện rung động!");
      digitalWrite(LED_PIN, HIGH);
      vibrationDetected = true;
      Blynk.virtualWrite(V1, 1);  // Báo rung động lên Blynk
      Blynk.virtualWrite(V2, accel);  // Gửi giá trị gia tốc
    }
  } else {
    if (vibrationDetected) {
      Serial.println("Không còn rung động.");
      digitalWrite(LED_PIN, LOW);
      vibrationDetected = false;
      Blynk.virtualWrite(V1, 0);  // Tắt báo rung động
      Blynk.virtualWrite(V2, accel);  // Cập nhật giá trị gia tốc
    }
  }
}
```

```cpp
// Gửi dữ liệu định kỳ lên Blynk
void sendBlynkData() {
  static unsigned long lastTime = 0;
  if (!IsReady(lastTime, 1000)) return;  // Gửi mỗi 1 giây
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  float accel = sqrt(a.acceleration.x * a.acceleration.x +
                     a.acceleration.y * a.acceleration.y +
                     a.acceleration.z * a.acceleration.z);
  Blynk.virtualWrite(V2, accel);  // Gửi giá trị gia tốc lên Blynk
}

void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);

  // Khởi tạo I2C và MPU6050
  Wire.begin();
  if (!mpu.begin()) {
    Serial.println("Không tìm thấy MPU6050!");
    while (1) delay(10);
  }
  Serial.println("MPU6050 đã sẵn sàng!");
  mpu.setAccelerometerRange(MPU6050_RANGE_2_G);

  // Kết nối WiFi và Blynk
  Serial.print("Connecting to ");
  Serial.println(ssid);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  Serial.println("WiFi connected");
  Serial.println("== START ==>");
}

void loop() {
  Blynk.run();  // Chạy Blynk
  currentMiliseconds = millis();
  updateVibration();  // Cập nhật trạng thái rung động
  sendBlynkData();    // Gửi dữ liệu lên Blynk
}

// Xử lý lệnh từ Blynk (nếu cần điều khiển LED từ ứng dụng)
BLYNK_WRITE(V1) {
  int value = param.asInt();
  if (value == 1) {
    Serial.println("Blynk -> Bật LED thủ công");
    digitalWrite(LED_PIN, HIGH);
  } else {
    Serial.println("Blynk -> Tắt LED thủ công");
    digitalWrite(LED_PIN, LOW);
  }
}
```

Hình ảnh code main.cpp

3.2 Chi tiết mã lập trình

<u>Đọc dữ liệu từ cảm biến MPU6050</u>

```cpp
// Khởi tạo MPU6050
Adafruit_MPU6050 mpu;
```

=> Khai báo đối tượng giao tiếp với cảm biến MPU6050

```
Wire.begin();
if (!mpu.begin()) {
  Serial.println("Không tìm thấy MPU6050!");
  while (1) delay(10);
}
Serial.println("MPU6050 đã sẵn sàng!");
mpu.setAccelerometerRange(MPU6050_RANGE_2_G);
```

=> Khởi tạo giao tiếp I2C (mặc định dùng SDA – GPIO21, SCL – GPIO22)

```
sensors_event_t a, g, temp;
mpu.getEvent(&a, &g, &temp);
```

=> **getEvent()** được gọi để đọc gia tốc 3 trục (x,y,z) và nhiệt độ hiện tại từ MPU6050

- Các dữ liệu này được sử dụng để tính toán độ rung

<u>Xử lý dữ liệu trên ESP32</u>

```
float accel = sqrt(a.acceleration.x * a.acceleration.x +
                   a.acceleration.y * a.acceleration.y +
                   a.acceleration.z * a.acceleration.z);
```

```
if (accel > threshold) {
  if (!vibrationDetected) {
    Serial.println("Phát hiện rung động!");
    digitalWrite(LED_PIN, HIGH);
    vibrationDetected = true;
  }
} else {
  if (vibrationDetected) {
    Serial.println("Không còn rung động.");
    digitalWrite(LED_PIN, LOW);
    vibrationDetected = false;
```

- Tính toán tổng gia tốc không gian 3 chiều bằng công thức " căn bậc hai của tổng bình phương " từng trục
- Ngưỡng threshold = 1.5 là mốc xác định thiết bị đang rung mạnh hay khôngvibrationDetected là biến trạng thái để tránh cảnh báo nhấp nhánh – chính là ngưỡng trễ (hysteresis)
- Nếu giá trị gia tốc vượt ngưỡng => Xác nhận rung => Bật cảnh báo
- Nếu rung thấp hơn và trạng thái đang rung trước đó => Tắt cảnh báo

<u>Điều khiển LED cảnh báo</u>

```
// Định nghĩa chân
#define LED_PIN 2  // Chân kết nối LED
```

```
void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
```

```
  digitalWrite(LED_PIN, HIGH);  digitalWrite(LED_PIN, LOW);
```

- LED_PIN kết nối đến chân GPIO của ESP32 (ở đây là GPIO2 theo mặc định trong code).
- Được bật khi phát hiện rung, và tắt khi không rung.
- Đèn LED hoạt động như một tín hiệu cảnh báo vật lý tại chỗ.

<u>Gửi dữ liệu lên Blynk để giám sát từ xa</u>

```
/* Thông tin Blynk */
#define BLYNK_TEMPLATE_ID "TMPL6Bmf8f_0b"
#define BLYNK_TEMPLATE_NAME "ESP32 MPU6050"
#define BLYNK_AUTH_TOKEN "uTYeUxybdAzm0-VZ6EgHlcj9Gc6tPPQv"
```

```
  Blynk.virtualWrite(V1, 1);  // Báo rung động lên Blynk
  Blynk.virtualWrite(V2, accel);  // Gửi giá trị gia tốc
}
```

```
  // Kết nối WiFi và Blynk
  Serial.print("Connecting to ");
  Serial.println(ssid);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  Serial.println("WiFi connected");
  Serial.println("== START ==>");
}
```

- ESP32 kết nối Wi-Fi qua ssid và pass, sau đó kết nối Blynk.
- V1 là widget cảnh báo (hiển thị LED cảnh báo trên app).
- V2 là widget hiển thị biểu đồ hoặc giá trị **gia tốc rung** theo thời gian thực.
- sendBlynkData() được gọi mỗi giây để cập nhật giá trị mới lên Blynk Cloud.

CHƯƠNG 4 : KẾT QUẢ VÀ ĐÁNH GIÁ

I. KẾT QUẢ THỬ NGHIỆM

Hệ thống đã được kiểm tra trên mô phỏng Wokwi và giám sát qua Blynk, cho kết quả như mong đợi:

• Khi độ rung vượt ngưỡng, LED cảnh báo sáng.
• Khi độ rung trở lại mức an toàn, LED tự động tắt.
• Dữ liệu rung hiển thị chính xác theo thời gian thực trên ứng dụng Blynk.
• Giao diện Blynk phản hồi nhanh, cho phép theo dõi liên tục tình trạng rung của thiết bị.

II. ĐÁNH GIÁ HIỆU SUẤT

• ***Ưu điểm:***
- Hệ thống hoạt động tự động và phản ứng nhanh với rung động.
- Có thể giám sát từ xa qua Internet nhờ tích hợp Blynk.
- Đơn giản, dễ triển khai, phù hợp với các ứng dụng cảnh báo rung trong công nghiệp.
- Tiết kiệm điện năng nhờ sử dụng vi điều khiển ESP32 hiệu suất cao.

• ***Nhược điểm:***
- Phụ thuộc vào kết nối Wi-Fi nếu muốn giám sát từ xa qua Blynk.
- LED chỉ cảnh báo bằng tín hiệu đơn giản, chưa tích hợp thêm loa, thông báo nâng cao hoặc điều khiển phản hồi khác.
- Không đo rung động theo đơn vị tiêu chuẩn (g-force) trong mô phỏng, cần bổ sung nếu triển khai thực tế.

III. ĐỀ XUẤT CẢI TIẾN

❖ Tích hợp thêm cảnh báo âm thanh: Bổ sung còi buzzer để đưa ra cảnh báo âm thanh khi phát hiện rung động vượt ngưỡng, giúp người dùng dễ dàng nhận biết hơn trong môi trường ồn ào hoặc không theo dõi Blynk thường xuyên.
❖ Sử dụng nhiều cảm biến tại các vị trí khác nhau: Việc sử dụng nhiều cảm biến MPU6050 ở các điểm khác nhau của thiết bị giúp phát hiện chính xác vị trí rung, từ đó đánh giá toàn diện tình trạng hoạt động.
❖ Lọc nhiễu tín hiệu cảm biến: Ứng dụng các thuật toán xử lý tín hiệu như bộ lọc Kalman hoặc trung bình trượt (moving average) để loại bỏ nhiễu và cho kết quả đo ổn định, chính xác hơn.
❖ Bổ sung chức năng ghi log dữ liệu: Lưu trữ dữ liệu độ rung theo thời gian vào thẻ nhớ hoặc gửi lên đám mây để phân tích lịch sử và dự đoán hỏng hóc theo xu hướng.

CHƯƠNG 5: HƯỚNG PHÁT TRIỂN TƯƠNG LAI

I. CẢI TIẾN PHẦN CỨNG

- Bổ sung còi báo động (Buzzer): Giúp người vận hành dễ dàng phát hiện sự cố rung bất thường thông qua tín hiệu âm thanh, nhất là trong môi trường không có màn hình hoặc khi không kết nối mạng.
- Tích hợp pin dự phòng (Li-ion hoặc pin sạc): Đảm bảo hệ thống vẫn hoạt động liên tục trong trường hợp mất điện, đặc biệt quan trọng ở các thiết bị công nghiệp hoặc vùng xa.
- Thêm nhiều cảm biến MPU6050: Triển khai nhiều cảm biến tại các vị trí khác nhau trên thiết bị/máy móc để giám sát rung toàn diện, phát hiện chính xác khu vực có sự cố.
- Tối ưu vỏ bảo vệ thiết bị: Thiết kế vỏ chống bụi, chống nước (theo chuẩn IP) giúp hệ thống hoạt động bền bỉ hơn trong môi trường công nghiệp khắc nghiệt.
- Tách mạch LED và tín hiệu điều khiển: Dùng opto cách ly hoặc transistor để bảo vệ ESP32 khỏi các xung điện áp khi điều khiển cảnh báo rung qua LED hoặc còi.

II. CẢI TIẾN PHẦN MỀM

- Áp dụng thuật toán lọc nhiễu (Filtering): Triển khai bộ lọc số (moving average, Kalman filter) để xử lý dữ liệu từ MPU6050, giảm nhiễu và tăng độ ổn định khi đo rung.
- Cài đặt nhiều mức cảnh báo: Thay vì chỉ một ngưỡng, có thể định nghĩa nhiều mức như: Rung nhẹ – Trung bình – Nguy hiểm, từ đó điều chỉnh mức cảnh báo tương ứng (LED nhấp nháy, còi liên tục, gửi cảnh báo khẩn cấp).
- Tích hợp tính năng lưu log dữ liệu: Ghi lại lịch sử độ rung theo thời gian vào bộ nhớ hoặc gửi về máy chủ để phục vụ việc phân tích, chẩn đoán sự cố theo thời gian.
- Gửi thông báo tự động qua Blynk hoặc email/SMS: Khi phát hiện rung động vượt ngưỡng nguy hiểm, hệ thống có thể tự động gửi cảnh báo đến người quản lý, không cần kiểm tra thủ công.
- Tùy chỉnh ngưỡng cảnh báo từ xa: Cho phép người dùng thay đổi ngưỡng rung cần cảnh báo ngay trên giao diện ứng dụng Blynk mà không cần phải nạp lại chương trình vào ESP32.

TÀI LIỆU THAM KHẢO

[1] Espressif Systems. (2023). ESP32 Technical Reference Manual. https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[2] Blynk Documentation. (2024). Getting Started with Blynk.

https://docs.blynk.io

[3] Wokwi. (2025). Wokwi Simulator - Online ESP32 and Sensor Simulator. https://wokwi.com

[4] InvenSense. (2021). MPU-6050 Product Specification. https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/

[5]Mô-đun gia tốc kế và con quay hồi chuyển

https://mecsu.vn/ho-tro-ky-thuat/modun-gia-toc-ke-mpu6050-va-con-quay-hoi-chuyen.9Jx

[6] Cảm biến gia tốc con quay hồi chuyển

https://www.laptrinhdientu.com/2022/08/MPU6050.html

PHẦN ĐÁNH GIÁ

TRƯỜNG ĐẠI HỌC KHOA HỌC HUẾ
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

PHIẾU ĐÁNH GIÁ TIỂU LUẬN

Học kỳ II Năm học 2024 - 2025

Cán bộ chấm thi 1	Cán bộ chấm thi 2
Nhận xét: .... **Điểm đánh giá của CBChT1:** Bằng số: ..... Bằng chữ:	**Nhận xét**: .... **Điểm đánh giá của CBChT2:** Bằng số:..... Bằng chữ: .....

Điểm kết luận: Bằng số.....Bằng chữ:.....

Thừa Thiên Huế, ngày 12 tháng 04 năm 2025

CBChT1

(Ký và ghi rõ họ tên)

CBChT2

(Ký và ghi rõ họ tên)