HỆ THỐNG THEO DÕI SỨC KHỎE THÔNG MINH (IOT)

Khoa Điện- Điện tử, Trường Cao Đẳng Công Thương Tp.HCM

NHD:Nguyễn Kim Suyên

TVTH:- TÂN NGỌC THANH CHÂU_MSSV:2123060038

- HOÀNG NGỌC BỘ_MSSV: 2123060025

1. GIỚI THIỆU

Trong bối cảnh công nghệ 4.0, việc ứng dụng IoT (Internet of Things) vào y tế đang trở thành xu hướng tất yếu. Hệ thống theo dõi sức khỏe từ xa giúp người dùng, đặc biệt là người già và người có bệnh nền, có thể tự giám sát các chỉ số sinh tồn cơ bản như nhịp tim, nồng độ oxy trong máu (SpO2) và trạng thái vận động hàng ngày. Báo cáo này trình bày giải pháp kết hợp giữa vi điều khiển **ESP32**, các cảm biến chuyên dụng và ứng dụng di động **Flutter**.

2. MỤC TIÊU ĐỀ TÀI

- **Giám sát thời gian thực:** Đo và hiển thị nhịp tim (BPM) và SpO2 (%).
- **Nhận diện vận động:** Sử dụng cảm biến gia tốc để phân tích trạng thái đi bộ, chạy hoặc đứng yên.
- **Cảnh báo khẩn cấp:** Tích hợp nút nhấn SOS và thuật toán phát hiện bất thường để kích hoạt còi hú/LED.
- **Số hóa dữ liệu:** Truyền dữ liệu không dây lên ứng dụng điện thoại để người thân dễ dàng theo dõi.

3. DANH MỤC THIẾT BỊ PHẦN CỨNG

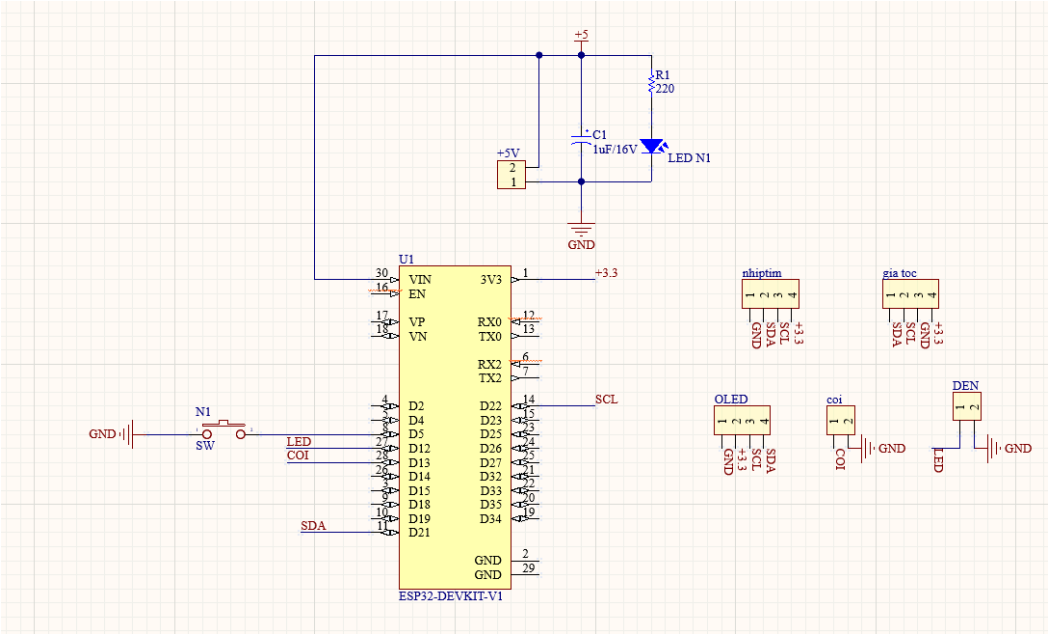
STT	Tên linh kiện	Chức năng chính
1	**ESP32 Dev Kit**	Trung tâm xử lý, kết nối Wi-Fi/Bluetooth.
2	**MAX30100**	Cảm biến đo nhịp tim và SpO2 qua hồng ngoại.
3	**MPU6050**	Cảm biến 6 trục (Gia tốc + Con quay hồi chuyển).

4	**OLED 0.96 inch**	Hiển thị thông số tại chỗ (Giao tiếp I2C).
5	**Buzzer & LED**	Phát tín hiệu cảnh báo âm thanh và ánh sáng.
6	**Nút nhấn (Button)**	Kích hoạt chế độ cứu hộ khẩn cấp (SOS).

---

4. THIẾT KẾ HỆ THỐNG

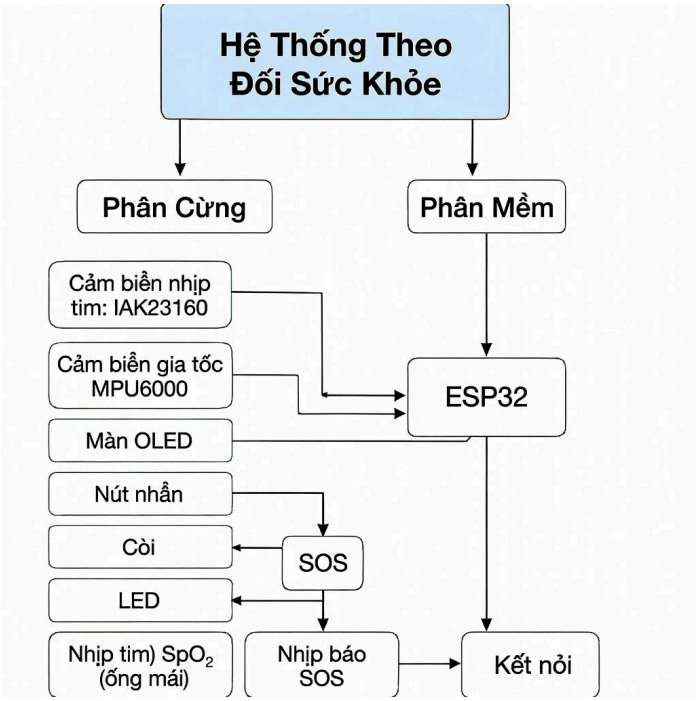
4.1. Sơ đồ kết nối (Wiring)



Hệ thống sử dụng chuẩn giao tiếp I2C để tối ưu số lượng chân cắm trên ESP32:

- **Chân SDA:** Kết nối với GPIO 21.
- **Chân SCL:** Kết nối với GPIO 22.
- **MAX30100/MPU6050/OLED:** Tất cả song song trên đường bus I2C này.
- **Nút nhấn SOS:** Kết nối GPIO 5 (Internal Pull-up).
- **Buzzer & LED:** Kết nối GPIO 13 và GPIO 12.

4.2 Sơ đồ khối



4.3. Nguyên lý hoạt động

1. **Thu thập:** ESP32 đọc dữ liệu thô từ MAX30100 và MPU6050 mỗi 500ms.
2. **Xử lý:** * Lọc nhiễu tín hiệu nhịp tim.
    - Tính toán vector gia tốc tổng $G_{total} = \sqrt{a_x^2 + a_y^2 + a_z^2}$ để xác định vận động.
3. **Hiển thị & Truyền tải:** Dữ liệu xuất ra màn hình OLED và đóng gói định dạng JSON để gửi qua WebSocket tới App Flutter.
4. **Phản hồi:** Nếu nhịp tim vượt ngưỡng hoặc nút SOS được nhấn, hệ thống ưu tiên ngắt để báo động.

5. PHẦN MỀM VÀ GIAO DIỆN

5.1. Chương trình trên ESP32

Viết trên nền tảng Arduino IDE, sử dụng các thư viện:

code esp32 :

```
#include <Wire.h>

#include <WiFi.h>

#include <WebSocketsServer.h>

#include <Adafruit_MPU6050.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>

#include <Preferences.h>          // Thư viện để lưu cài đặt vào bộ nhớ Flash (không mất khi tắt nguồn)
```

```
#include "MAX30100_PulseOximeter.h"

/* ================== CẤU HÌNH PHẦN CỨNG ================== */
#define BUTTON_PIN   5            // Nút nhấn SOS nối chân số 5
#define LED_PIN      12           // LED báo hiệu nối chân số 12
#define BUZZER_PIN   13           // Còi báo động nối chân số 13
#define SCREEN_WIDTH 128          // Chiều rộng màn hình OLED
#define SCREEN_HEIGHT 64          // Chiều cao màn hình OLED

// Khởi tạo các đối tượng điều khiển
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1); // Điều khiển OLED
WebSocketsServer webSocket(80);                    // Tạo server WebSocket cổng 80
Adafruit_MPU6050 mpu;                              // Cảm biến gia tốc/ngã
PulseOximeter pox;                                 // Cảm biến nhịp tim/SpO2
Preferences pref;                                  // Đối tượng quản lý bộ nhớ Flash

// Thông số WiFi (ESP32 phát WiFi cho điện thoại kết nối)
const char *ssid = "ESP32_TheoDoiSucKhoe";
const char *password = "12345678";

/* ================== BIẾN CÀI ĐẶT (ĐỒNG BỘ VỚI APP) ================== */
float hrMax = 120.0, hrMin = 50.0, spo2Min = 94.0;          // Ngưỡng sức khỏe
bool isFallDetectionEnabled = true;                         // Bật/tắt báo ngã
float thresStand = 1.0, thresWalk = 2.5, thresRun = 5.0, thresFall = 8.0; // Ngưỡng lực G
int earlyWarning = 1, leadTime = 5, sensitivity = 10;       // Các biến dự phòng cho App

// Biến quản lý thời gian và trạng thái
uint32_t lastDataMillis = 0, lastBlinkMillis = 0, lastOLEDMillis = 0;
bool oledReady = false, mpuReady = false, poxReady = false;
```

```
bool alertState = false, fingerDetected = false;

/* ================== QUẢN LÝ BỘ NHỚ FLASH ================== */
// Hàm tải dữ liệu cũ từ bộ nhớ lên các biến khi vừa khởi động
void loadConfig() {
  pref.begin("health_cfg", true); // Mở vùng nhớ "health_cfg" ở chế độ Read-only
  hrMax = pref.getFloat("hrMax", 120.0);
  hrMin = pref.getFloat("hrMin", 50.0);
  spo2Min = pref.getFloat("spo2Min", 94.0);
  isFallDetectionEnabled = pref.getBool("fallEn", true);
  thresFall = pref.getFloat("tFall", 8.0);
  pref.end();
  Serial.println("[SYSTEM] Đã tải cài đặt từ bộ nhớ Flash");
}

// Hàm lưu dữ liệu mới từ App xuống bộ nhớ Flash
void saveConfig() {
  pref.begin("health_cfg", false); // Mở vùng nhớ ở chế độ Ghi (false)
  pref.putFloat("hrMax", hrMax);
  pref.putFloat("hrMin", hrMin);
  pref.putFloat("spo2Min", spo2Min);
  pref.putBool("fallEn", isFallDetectionEnabled);
  pref.putFloat("tFall", thresFall);
  pref.end();
  Serial.println("[SYSTEM] Đã lưu cài đặt mới vào Flash");
}

/* ================== XỬ LÝ GIAO TIẾP VỚI APP ================== */
// Hàm tách chuỗi từ App gửi xuống (ví dụ: "SET_CONFIG:120,50,94...")
```

```
void handleCommand(String msg) {
  if (msg.startsWith("SET_CONFIG:")) {
    String data = msg.substring(11); // Cắt bỏ chữ "SET_CONFIG:"

    int i = 0;
    String v[11]; // Mảng chứa 11 thông số cắt được
    int start = 0, end = data.indexOf(',');

    // Vòng lặp cắt chuỗi dựa trên dấu phẩy
    while (end != -1 && i < 10) {
      v[i++] = data.substring(start, end);
      start = end + 1;
      end = data.indexOf(',', start);
    }
    v[i] = data.substring(start); // Lấy phần tử cuối cùng

    // Chuyển chuỗi thành số và cập nhật vào biến hệ thống
    hrMax = v[0].toFloat();
    hrMin = v[1].toFloat();
    spo2Min = v[2].toFloat();
    earlyWarning = v[3].toInt();
    leadTime = v[4].toInt();
    sensitivity = v[5].toInt();
    isFallDetectionEnabled = (v[6] == "1");
    thresStand = v[7].toFloat();
    thresWalk = v[8].toFloat();
    thresRun = v[9].toFloat();
    thresFall = v[10].toFloat();
```

```
    saveConfig(); // Lưu lại vào Flash để lần sau khởi động vẫn còn
    Serial.println("[APP] Cập nhật cấu hình thành công");
  }
}

// Sự kiện khi có kết nối WebSocket
void onWebSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {
  if (type == WStype_TEXT) {
    handleCommand((char*)payload); // Chuyển tin nhắn nhận được vào hàm xử lý
  }
}

/* ================== KHỞI TẠO HỆ THỐNG ================== */
void setup() {
  Serial.begin(115200);
  pinMode(BUTTON_PIN, INPUT_PULLUP); // Chân nút nhấn dùng điện trở kéo lên
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);

  Wire.begin(21, 22); // Khởi tạo I2C cho OLED và cảm biến

  // Kiểm tra kết nối các linh kiện
  oledReady = display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  mpuReady = mpu.begin();
  poxReady = pox.begin();

  if (poxReady) pox.setIRLedCurrent(MAX30100_LED_CURR_4_4MA); // Chỉnh cường độ tia IR

  loadConfig(); // Bước quan trọng: Lấy lại các cài đặt cũ
```

```
  WiFi.softAP(ssid, password); // Phát WiFi
  webSocket.begin();
  webSocket.onEvent(onWebSocketEvent);

  Serial.println("[SYSTEM] ESP32 Ready!");
}

/* ================== VÒNG LẶP CHÍNH ================== */
void loop() {
  webSocket.loop();    // Duy trì kết nối với App
  if (poxReady) pox.update(); // Cập nhật dữ liệu từ cảm biến nhịp tim liên tục

  bool isEmergency = false; // Biến đánh dấu có sự cố hay không
  float gForce = 0;
  String motion = "DUNG YEN";

  // 1. Xử lý dữ liệu MPU6050 (Gia tốc & Ngã)
  if (mpuReady && isFallDetectionEnabled) {
    sensors_event_t a, g, t;
    mpu.getEvent(&a, &g, &t);
    // Công thức tính lực G tổng hợp: $G = \sqrt{a_x^2 + a_y^2 + a_z^2}/9.81$
    gForce = sqrt(sq(a.acceleration.x) + sq(a.acceleration.y) + sq(a.acceleration.z)) / 9.81;
    if (gForce > thresFall) { motion = "NGA"; isEmergency = true; }
    else if (gForce > thresRun) motion = "CHAY";
    else if (gForce > thresWalk) motion = "DI BO";
  }
  // 2. Xử lý dữ liệu nhịp tim & Oxy
  float hr = poxReady ? pox.getHeartRate() : 0;
```

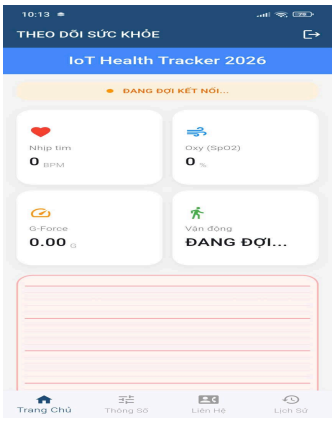
```
float spo2 = poxReady ? pox.getSpO2() : 0;
  // Kiểm tra xem có áp ngón tay vào cảm biến không
fingerDetected = (hr > 30 && spo2 > 80);
// 3. Kiểm tra các điều kiện khẩn cấp
if (digitalRead(BUTTON_PIN) == LOW) { isEmergency = true; motion = "SOS"; } // Nhấn nút SOS
if (fingerDetected && (hr > hrMax || hr < hrMin || spo2 < spo2Min)) {
  isEmergency = true; // Sức khỏe vượt ngưỡng cài đặt
}
// 4. Thực thi cảnh báo (LED & Còi)
if (isEmergency) {
  if (millis() - lastBlinkMillis > 300) { // Chớp tắt mỗi 300ms
    lastBlinkMillis = millis();
    alertState = !alertState;
    digitalWrite(LED_PIN, alertState);
    digitalWrite(BUZZER_PIN, alertState);
    // Gửi tín hiệu ALERT về điện thoại
    if (alertState) webSocket.broadcastTXT("{\"type\":\"ALERT\",\"level\":\"SOS\"}");
  }
} else {
  // Nếu không khẩn cấp, đèn LED chỉ sáng khi có điện thoại kết nối
  digitalWrite(LED_PIN, WiFi.softAPgetStationNum() > 0);
  digitalWrite(BUZZER_PIN, LOW);
}

// 5. Định kỳ gửi dữ liệu JSON về App (0.5 giây/lần)
if (millis() - lastDataMillis >= 500) {
  lastDataMillis = millis();
  String json;
  if (fingerDetected) {
```

```
      json = "{\"type\":\"DATA\",\"hr\":" + String(hr,1) + ",\"spo2\":" + String(spo2,0) +
             ",\"g\":" + String(gForce,2) + ",\"motion\":\"" + motion + "\"}";
    } else {
      json = "{\"type\":\"NO_FINGER\"}"; // Báo cho App là chưa đặt tay
    }
    webSocket.broadcastTXT(json);
  }
  // 6. Hiển thị thông số lên màn hình OLED
  if (oledReady && millis() - lastOLEDMillis >= 500) {
    lastOLEDMillis = millis();
    display.clearDisplay();
    display.setTextColor(SSD1306_WHITE);
    if (isEmergency && alertState) {
      display.setTextSize(2); display.setCursor(30, 20); display.println("SOS!");
    } else {
      display.setTextSize(1);
      display.setCursor(0, 0);
      display.printf("HR  : %.1f bpm\n", hr);
      display.printf("SpO2: %.0f %%\n", spo2);
      display.printf("G   : %.2f\n", gForce);
      display.printf("STT : %s", motion.c_str());
    }
    display.display();
  }
}
```

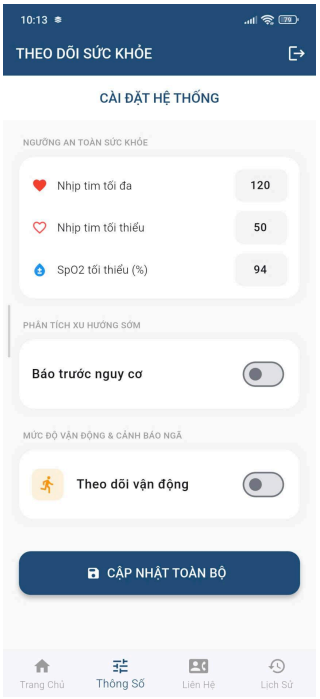
- MAX30100_PulseOximeter.h: Xử lý sinh hiệu.
- Adafruit_MPU6050.h: Đọc gia tốc.
- WebSocketsServer.h: Truyền dữ liệu thời gian thực.

5.2. Ứng dụng Flutter

-Trang chủ: Hiển thị Dashboard với các thẻ thông số (Card) và biểu đồ nhịp tim động.



-Trang cài đặt: Cho phép tùy chỉnh ngưỡng cảnh báo (ví dụ: báo động khi nhịp tim > 120 BPM).



-Thông báo: Hiển thị Dialog cảnh báo đỏ khi nhận được tín hiệu SOS từ phần cứng.

6. KẾT QUẢ VÀ ĐÁNH GIÁ

6.1. Kết quả đạt được

- Hệ thống khởi động nhanh, kết nối App ổn định trong môi trường Wi-Fi nội bộ.
- Cảm biến MAX30100 đo chính xác khi người dùng đứng yên.
- Tính năng SOS phản hồi gần như lập tức (<100ms).

6.2. Đánh giá

- **Ưu điểm:** Giá thành rẻ, tính ứng dụng cao, giao diện App hiện đại.
- **Hạn chế:** +Cảm biến MAX30100 dễ bị nhiễu khi người dùng di chuyển mạnh (cần cải thiện thuật toán lọc nhiễu bù trừ gia tốc).

7. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Dự án đã hoàn thành các mục tiêu đề ra, tạo được một mô hình giám sát sức khỏe cơ bản nhưng hiệu quả. Trong tương lai, hệ thống có thể tích hợp thêm **cảm biến nhiệt độ hồng ngoại** và đẩy dữ liệu lên **Firebase Cloud** để lưu trữ lịch sử sức khỏe theo tháng/năm.