**🧾 BÁO CÁO: CHƯƠNG TRÌNH ESP32 NHẬN DỮ LIỆU BẰNG GIAO THỨC ESP-NOW**

**1️⃣. Mục đích thí nghiệm**

Xây dựng chương trình cho vi điều khiển **ESP32** có khả năng **nhận dữ liệu không dây từ một ESP32 khác** thông qua **giao thức ESP-NOW** mà **không cần sử dụng Wi-Fi Router** hay mạng Internet.

**2️⃣. Giới thiệu giao thức ESP-NOW**

**ESP-NOW** là giao thức truyền thông không dây do **Espressif** phát triển, cho phép các thiết bị ESP32/ESP8266 **trao đổi dữ liệu trực tiếp với nhau** (peer-to-peer).

* Giao tiếp **nhanh, ổn định, độ trễ thấp (~3 ms)**.
* Không cần Wi-Fi Access Point.
* Có thể gửi dữ liệu tới **nhiều thiết bị cùng lúc (broadcast)**.
* Phù hợp cho **ứng dụng cảm biến, IoT, SCADA, điều khiển phân tán, v.v.**

**3️⃣. Phần cứng sử dụng**

| **Thiết bị** | **Số lượng** | **Ghi chú** |
| --- | --- | --- |
| ESP32 (hoặc LOLIN C3 Mini) | 1 | Module nhận dữ liệu |
| Dây USB Type-C | 1 | Dùng để nạp chương trình và giám sát Serial |
| Máy tính cài Arduino IDE | 1 | Biên dịch và nạp code |

**4️⃣. Sơ đồ kết nối**

ESP32 nhận không cần kết nối ngoại vi (vì chỉ xử lý dữ liệu qua sóng RF).  
Chỉ cần kết nối **ESP32 với máy tính qua cáp USB** để xem dữ liệu nhận trên **Serial Monitor**.

**5️⃣. Chương trình nhận dữ liệu**

#include <WiFi.h>

#include <esp\_now.h>

// ===============================

// ⚙️ Cấu trúc dữ liệu sẽ nhận

// ===============================

typedef struct struct\_message {

float temperature; // Nhiệt độ (°C)

float humidity; // Độ ẩm (%)

} struct\_message;

struct\_message incomingData; // Biến chứa dữ liệu nhận được

// ==================================================

// 📥 Hàm callback được gọi tự động khi có dữ liệu đến

// ==================================================

void OnDataRecv(const esp\_now\_recv\_info \*info, const uint8\_t \*data, int len) {

// Sao chép dữ liệu nhận được vào biến incomingData

memcpy(&incomingData, data, sizeof(incomingData));

// In ra màn hình Serial để kiểm tra

Serial.printf("📩 Nhận dữ liệu: %.2f °C, %.2f %%RH\n",

incomingData.temperature, incomingData.humidity);

}

// ===============================

// 🚀 Hàm setup() chạy 1 lần khi ESP khởi động

// ===============================

void setup() {

Serial.begin(115200); // Bật Serial để in thông tin

WiFi.mode(WIFI\_STA); // Đặt ESP vào chế độ Station

// Khởi tạo giao thức ESP-NOW

if (esp\_now\_init() != ESP\_OK) {

Serial.println("❌ ESP-NOW init thất bại!");

return;

}

// Đăng ký hàm callback khi có dữ liệu đến

esp\_now\_register\_recv\_cb(OnDataRecv);

Serial.println("✅ ESP nhận sẵn sàng!");

Serial.print("📡 MAC của ESP nhận: ");

Serial.println(WiFi.macAddress()); // In MAC để ESP gửi biết địa chỉ đích

}

void loop() {

delay(2000); // Giảm tải CPU

}

**6️⃣. Giải thích hoạt động**

| **Bước** | **Mô tả** |
| --- | --- |
| 1 | ESP32 được cấu hình ở chế độ **Station (Wi-Fi STA)**. |
| 2 | **ESP-NOW** được khởi tạo để bắt đầu cơ chế truyền-nhận dữ liệu. |
| 3 | Khi ESP khác gửi gói tin, hàm **OnDataRecv()** tự động được gọi. |
| 4 | Dữ liệu nhận (nhiệt độ, độ ẩm) được sao chép vào cấu trúc incomingData. |
| 5 | Thông tin được in ra màn hình Serial để quan sát kết quả. |

**7️⃣. Kết quả thực nghiệm**

* Khi ESP gửi dữ liệu, Serial Monitor của ESP nhận hiển thị:
* 📩 Nhận dữ liệu: 28.70 °C, 63.20 %RH
* 📩 Nhận dữ liệu: 29.10 °C, 61.90 %RH
* Tốc độ cập nhật trung bình: 1–2 giây/lần.
* Khoảng cách hoạt động ổn định: ~15–20 m trong nhà.

**8️⃣. Kết luận**

* Chương trình đã hoạt động **ổn định, nhận dữ liệu chính xác**.
* ESP-NOW giúp **truyền dữ liệu nhanh, tiêu thụ năng lượng thấp**, không phụ thuộc Wi-Fi mạng ngoài.
* Đây là **giải pháp hiệu quả** trong các ứng dụng như:
  + Truyền dữ liệu cảm biến giữa các ESP32.
  + Giao tiếp giữa các node trong mạng cảm biến IoT.
  + Truyền tín hiệu điều khiển trong hệ thống SCADA nhỏ.