TRƯỜNG ĐẠI HỌC KHOA HỌC HUẾ

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TIỂU LUẬN

PHÁT TRIỂN ỨNG DỤNG IOT

Đề tài: Xây dựng đồng hồ thời gian thực với ESP32 và NTP

Giáo viên hướng dẫn: Võ Việt Dũng

Huế, Ngày 30 Tháng 3 Năm 2025

MỤC LỤC

Lời mở đầu

Trong thời đại công nghệ số phát triển mạnh mẽ, việc tự động hóa và đồng bộ thời gian chính xác trong các hệ thống nhúng ngày càng trở nên quan trọng. Các ứng dụng như đồng hồ điện tử, hệ thống điều khiển thiết bị theo lịch, giám sát môi trường, hoặc hệ thống cảnh báo đều cần thời gian thực để vận hành hiệu quả. Tuy nhiên, việc duy trì thời gian chính xác trên các thiết bị nhúng thường gặp khó khăn do giới hạn về phần cứng, độ lệch thời gian hoặc thiếu bộ RTC (Real Time Clock).

Với sự ra đời của các vi điều khiển có kết nối Wi-Fi như **ESP32**, việc truy cập Internet để lấy thời gian từ các máy chủ NTP (Network Time Protocol) đã trở nên dễ dàng và chính xác hơn bao giờ hết. Nhờ đó, các hệ thống nhúng không cần đồng hồ phần cứng vẫn có thể cập nhật và duy trì thời gian chính xác theo thời gian thực.

Đề tài **"Xây dựng đồng hồ thời gian thực với ESP32 và NTP"** được thực hiện với mục tiêu thiết kế một hệ thống đơn giản, chi phí thấp nhưng có khả năng cập nhật và hiển thị thời gian theo giờ quốc tế (UTC) hoặc giờ địa phương, thông qua kết nối Internet. Hệ thống sử dụng **ESP32** làm bộ xử lý trung tâm, giao tiếp với máy chủ NTP để lấy thời gian thực và hiển thị trên màn hình **OLED** . Đề tài không chỉ mang tính thực tiễn cao, mà còn là nền tảng để phát triển các ứng dụng nâng cao như lịch hẹn giờ, hệ thống giám sát thời gian thực, hoặc điều khiển thiết bị theo lịch định sẵn.

Qua việc nghiên cứu và triển khai đề tài này, sinh viên có thể củng cố kiến thức về IoT, lập trình vi điều khiển, giao tiếp I2C, làm việc với thời gian thực và đặc biệt là khả năng tích hợp phần mềm – phần cứng trong các ứng dụng nhúng hiện đại.

I.Mục tiêu, phạm vi nghiên cứu.

1. Mục tiêu nghiên cứu
    - Thiết kế và xây dựng một hệ thống đồng hồ thời gian thực sử dụng ESP32, có khả năng lấy và cập nhật thời gian tự động thông qua giao thức NTP.
    - Hiển thị thời gian thực (giờ, phút, giây) lên màn hình OLED sử dụng giao tiếp I2C.
    - Đồng bộ thời gian theo múi giờ địa phương, có thể tùy chỉnh phù hợp với người sử dụng.
    - Làm nền tảng để phát triển các ứng dụng mở rộng như: đồng hồ báo thức, hệ thống điều khiển thiết bị theo lịch, thông báo theo thời gian,…
2. Phạm vi nghiên cứu
    - Chỉ tập trung vào việc xây dựng hệ thống đồng hồ thời gian thực sử dụng vi điều khiển ESP32 và giao thức NTP (Network Time Protocol) để đồng bộ thời gian từ Internet.
    - Phạm vi nghiên cứu xoay quanh nguyên lý hoạt động của giao thức NTP, cách thiết lập kết nối Wi-Fi trên ESP32, và quá trình lấy / xử lý /hiển thị thời gian thực.
    - Hệ thống được thử nghiệm với các chức năng cơ bản như: kết nối mạng Wi-Fi**,** lấy thời gian từ máy chủ NTP, và hiển thị thời gian lên màn hình OLED thông qua giao tiếp I2C.

II. RTC là gì?

RTC (Real-Time Clock) là một thành phần quan trọng giúp theo dõi thời gian thực mà không bị ảnh hưởng bởi việc mất điện hoặc khởi động lại hệ thống. RTC thường được sử dụng trong các ứng dụng cần quản lý thời gian như đồng hồ, bộ đếm thời gian, hoặc lịch**.**

Một số loại RTC phổ biến:

1. RTC tích hợp trong vi điều khiển:

-Một số vi điều khiển, như ESP32 hoặc STM32, có tích hợp sẵn RTC trong chip. Loại này thường được dùng khi cần đơn giản hoá thiết kế và giảm chi phí.

-Độ chính xác của RTC tích hợp phụ thuộc vào xung nhịp bên trong và thường kém hơn các RTC chuyên dụng.

2. RTC ngoài (RTC độc lập):

-Các mạch RTC ngoài như DS1307, DS3231 là những IC chuyên dụng chỉ làm nhiệm vụ quản lý thời gian. Chúng có thể hoạt động với pin dự phòng, nên khi mất nguồn chính vẫn duy trì được thời gian chính xác.

-DS3231 là một trong những loại phổ biến nhất với độ chính xác cao, còn DS1307 thường kém chính xác hơn.

3. RTC trong các thiết bị lưu trữ (như máy tính hoặc hệ thống nhúng lớn):

-Một số hệ thống lưu trữ lớn như máy tính hoặc hệ thống nhúng phức tạp có tích hợp sẵn RTC. Chúng sử dụng pin CMOS để duy trì thời gian khi tắt nguồn.

4. RTC trong FPGA (Field-Programmable Gate Array):

- Trong các hệ thống FPGA, đôi khi người ta cần thiết kế riêng module RTC để tích hợp vào ứng dụng. Tuy nhiên, đây là loại ít phổ biến hơn và thường gặp trong các dự án đòi hỏi tuỳ biến cao.

5. RTC trong các thiết bị IoT:RTC trong các thiết bị IoT:

- Các thiết bị IoT thường có RTC tích hợp để đồng bộ thời gian từ mạng (NTP – Network Time Protocol) khi kết nối mạng, và sử dụng RTC để giữ thời gian khi mất kết nối.

III. Cách dùng RTC trên ESP32

```
#include <time.h>

// Hàm thiết lập thời gian bắt đầu
void setupTime() {
  struct tm t;
  t.tm_year = 2024 - 1900; // Năm 2024 (tính từ 1900)
  t.tm_mon = 10; // Tháng 11 (tháng bắt đầu từ 0, nên 10 là tháng 11)
  t.tm_mday = 15; // Ngày 15
  t.tm_hour = 8; // Giờ 8
  t.tm_min = 30; // Phút 30
  t.tm_sec = 0; // Giây 0
  t.tm_isdst = 0; // Không có giờ mùa hè
  time_t epoch = mktime(&t); // Chuyển đổi struct tm thành time_t (giây từ Epoch)
  struct timeval tv = {epoch, 0}; // Tạo struct timeval
  settimeofday(&tv, NULL); // Thiết lập thời gian hệ thống
}
void setup() {
  Serial.begin(115200); // Khởi động Serial với tốc độ truyền 115200 bps
  setupTime(); // Gọi hàm thiết lập thời gian ban đầu
}
void loop() {
  time_t now;
  struct tm timeinfo;
```

```
  time(&now); // Lấy thời gian hiện tại
  localtime_r(&now, &timeinfo); // Chuyển đổi thời gian hiện tại thành struct tm
  // In thời gian hiện tại ra Serial Monitor
  Serial.println(&timeinfo, "%A, %d-%m-%Y %H:%M:%S");
  delay(1000); // Đợi 1 giây
}
```

Khi sử dụng thư viện **time.h**, chúng ta có một biến cấu trúc **tm** được định nghĩa, biến này được dùng để lưu giữ thông tin về thời gian. Các phần tử của biến cấu trúc **tm** bao gồm:

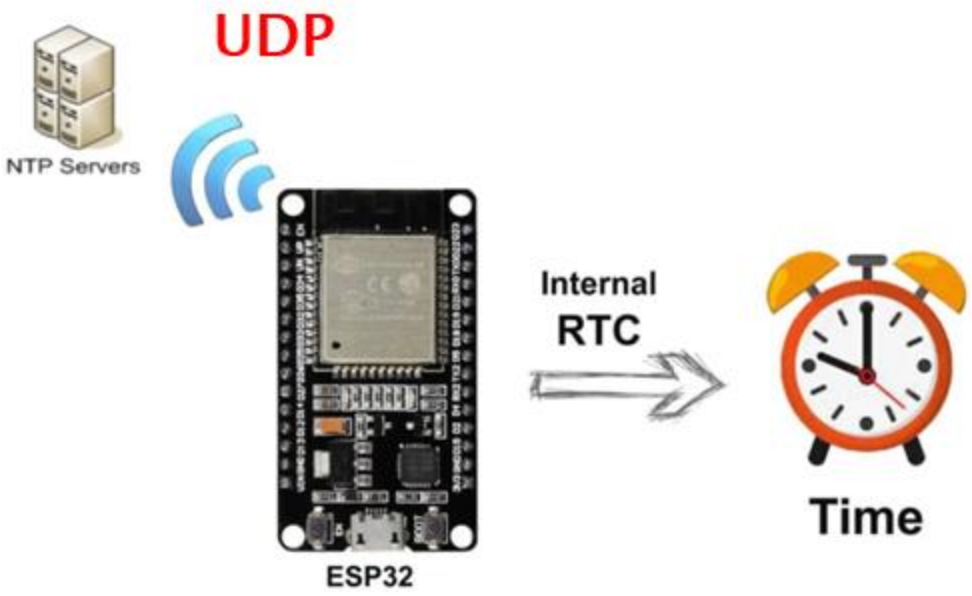
- **tm_sec**: Giây (số nguyên từ 0 đến 59).
- **tm_min**: Phút (số nguyên từ 0 đến 59).
- **tm_hour**: Giờ (số nguyên từ 0 đến 23).
- **tm_mday**: Ngày trong tháng (số nguyên từ 1 đến 31).
- **tm_mon**: Tháng trong năm (số nguyên từ 0 đến 11, với 0 là tháng Giêng và 11 là tháng Mười Hai).
- **tm_year**: Năm kể từ 1900 (ví dụ: 120 đại diện cho năm 2020).
- **tm_wday**: Ngày trong tuần (số nguyên từ 0 đến 6, với 0 là Chủ nhật và 6 là Thứ Bảy).
- **tm_yday**: Ngày trong năm (số nguyên từ 0 đến 365, với 0 là ngày 1 tháng 1).
- **tm_isdst**: Cờ xác định thời gian mùa hè (số dương nếu đang trong thời gian mùa hè, số 0 nếu không, và số âm nếu trạng thái này không xác định).

  Các ký tự **%** trong lệnh **Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S")** đều là phần của chuỗi định dạng thời gian. Chúng thuộc về thư viện **time.h** và là cách định dạng thời gian trong ngôn ngữ lập trình C. Những ký tự này được dùng để hiển thị các thành phần cụ thể của thời gian. Sau đây là giải thích ý nghĩa của mỗi ký tự:

- **%a**: Tên viết tắt của ngày trong tuần (ví dụ: "Mon").
- **%A**: Tên đầy đủ của ngày trong tuần (ví dụ: "Monday").
- **%b**: Tên viết tắt của tháng (ví dụ: "Jan").
- **%B**: Tên đầy đủ của tháng (ví dụ: "January").
- **%d**: Ngày trong tháng (số có 2 chữ số, ví dụ: "01" đến "31").
- **%H**: Giờ trong ngày (24 giờ, số có 2 chữ số, ví dụ: "00" đến "23").
- **%I**: Giờ trong ngày (12 giờ, số có 2 chữ số, ví dụ: "01" đến "12").
- **%m**: Tháng trong năm (số có 2 chữ số, ví dụ: "01" đến "12").
- **%M**: Phút trong giờ (số có 2 chữ số, ví dụ: "00" đến "59").
- **%p**: Chu kỳ AM hoặc PM.
- **%S**: Giây trong phút (số có 2 chữ số, ví dụ: "00" đến "59").
- **%y**: Năm dưới dạng số có hai chữ số (ví dụ: "22" cho năm 2022).
- **%Y**: Năm đầy đủ (ví dụ: "2024").

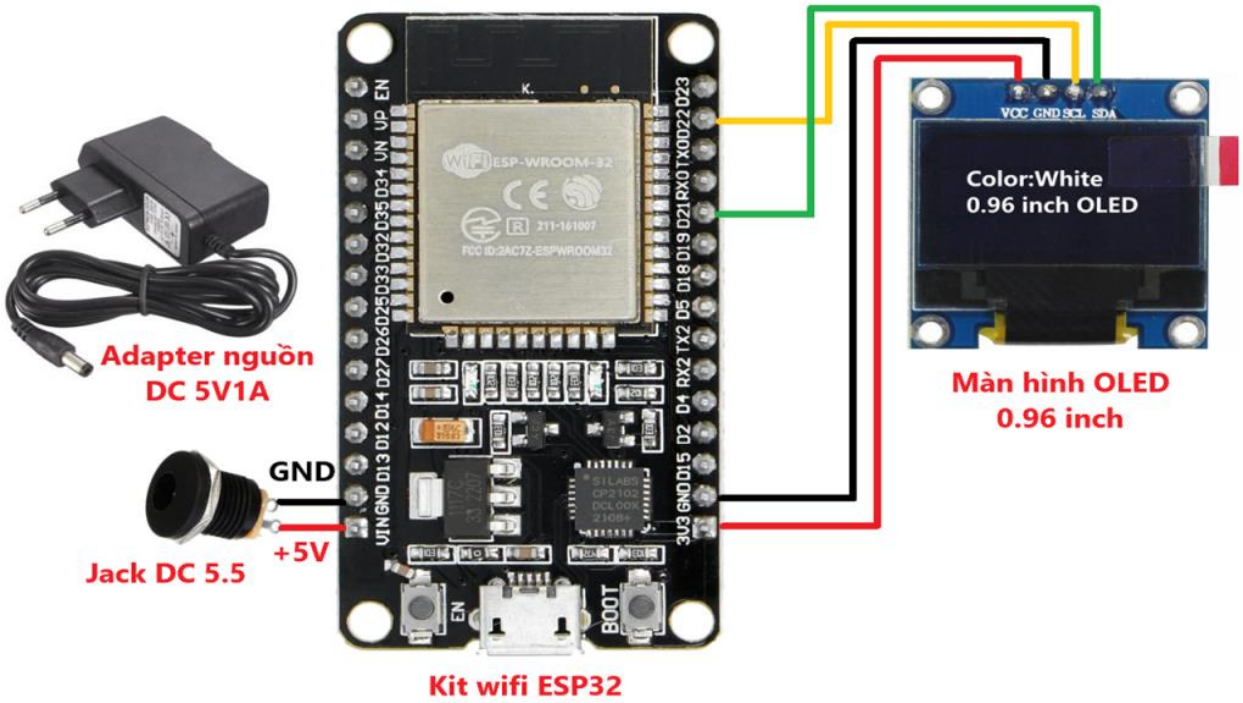
IV. Thiết kế mạch IoT

1. Sơ đồ logic của hệ thống



Đồng bộ thời gian với NTP cho RTC trên ESP32 mang lại nhiều lợi ích. Nó đảm bảo độ chính xác cao và tự động điều chỉnh sai số của RTC, giúp đơn giản hóa việc quản lý thời gian mà không cần thiết lập thủ công. Ngoài ra, khi ESP32 được đồng bộ hóa với NTP, nó có thể phục vụ như một nguồn thời gian chính xác cho các thiết bị khác trong mạng. Điều này đảm bảo tất cả các thiết bị đều có cùng một thời gian chuẩn, đặc biệt hữu ích cho các ứng dụng yêu cầu sự đồng bộ thời gian chặt chẽ.

2. Đấu nối phần cứng



STT	Tên linh kiện	Chức năng chính
1	ESP32 DevKit V1	Vi điều khiển tích hợp Wi-Fi, điều khiển toàn bộ hệ thống
2	Màn hình OLED 0.96 inch	Hiển thị thời gian thực (giờ, phút, giây)
3	Adapter nguồn 5V – 1A	Cấp nguồn ổn định cho mạch ngoài môi trường mô phỏng
4	Jack DC 5.5mm	Cổng cấp nguồn 5V từ Adapter vào mạch ESP32
5	Dây nối (jumper)	Kết nối nguồn và tín hiệu giữa các linh kiện

3. Nguyên lý hoạt động của hệ thống

-ESP32 khởi động, kết nối với Wi-Fi bằng SSID và mật khẩu đã lập trình sẵn.

-Nếu kết nối thành công, hệ thống sẽ tiếp tục lấy thời gian từ NTP.

-Sử dụng thư viện NTPClient hoặc time.h (tùy cách cài đặt).

-ESP32 gửi yêu cầu đến máy chủ NTP (ví dụ: pool.ntp.org).

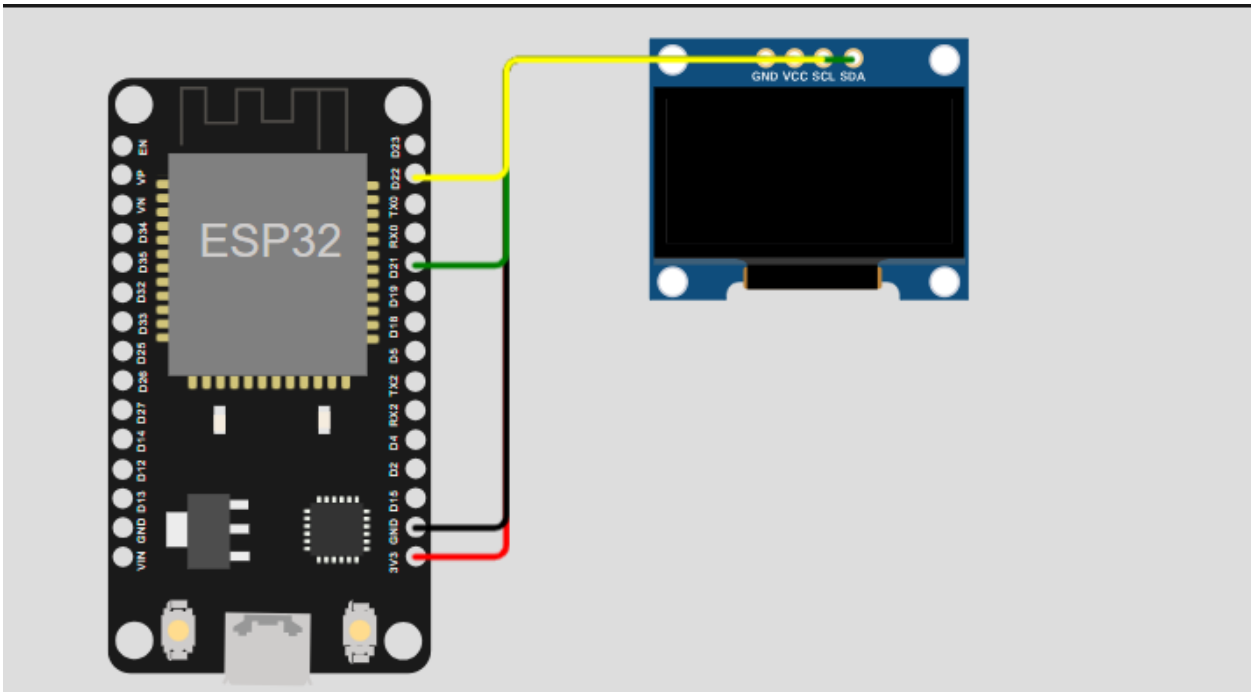
-Nhận thời gian UTC (Universal Time Coordinated).

-ESP32 xử lý chuỗi thời gian: tách giờ, phút, giây.

-Dữ liệu được gửi đến màn hình OLED qua giao thức I2C.

-Màn hình OLED sẽ liên tục cập nhật và hiển thị thời gian thực.

4. Mạch vật lý



V. Cài đặt hệ thống

1. Công cụ sử dụng

-Phần mềm lập trình: Arduino IDE

-Thư viện cần cài:

+Adafruit_GFX.h và Adafruit_SSD1306.h (OLED)

+WiFi.h (kết nối mạng)

+time.h hoặc NTPClient.h (lấy giờ NTP)

-Mô phỏng : Wokwi.com

-Công cụ nạp: Cáp Micro USB

2.Quy trình cài đặt

-Cài đặt Arduino IDE và thêm board ESP32 từ Board Manager.

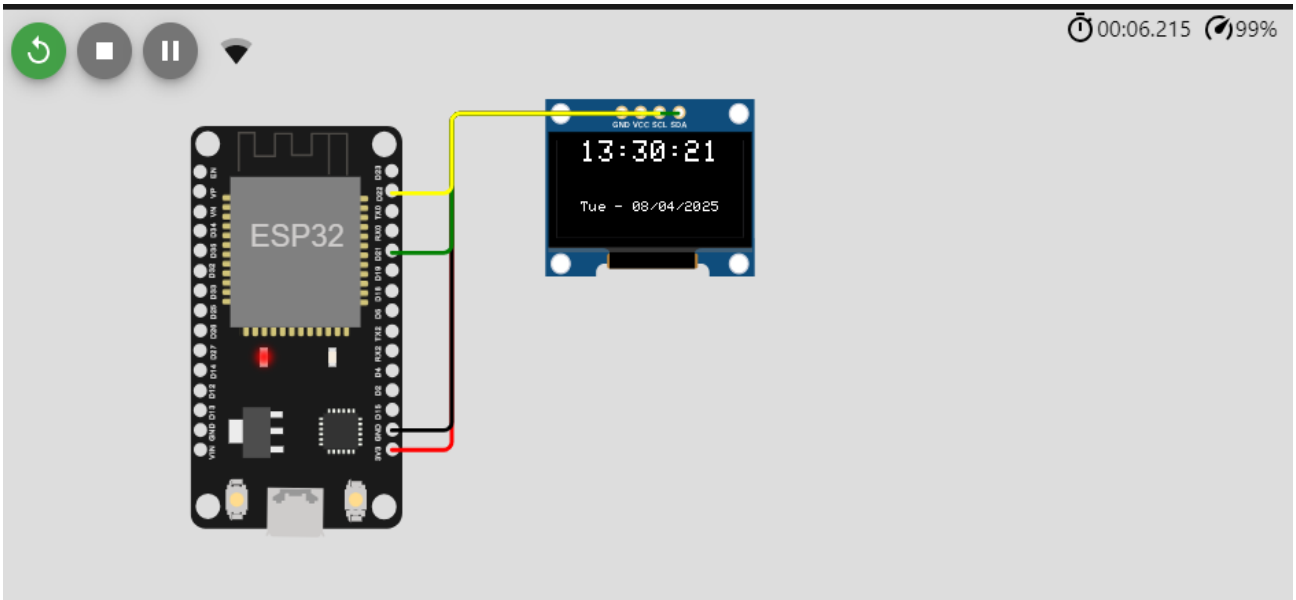
-Chọn board “ESP32 Dev Module”, cổng COM phù hợp.

-Kết nối ESP32 qua USB, nạp đoạn code thực hiện:

- Kết nối Wi-Fi
- Lấy thời gian từ máy chủ NTP
- Hiển thị lên OLED

-Quan sát kết quả trên màn hình OLED hoặc Serial Monitor.

VI.Kết quả thực hiện



Hoạt động của các linh kiện:

-ESP32 đóng vai trò trung tâm, kết nối Wi-Fi, lấy thời gian từ Internet và xử lý dữ liệu.

-Màn hình OLED sử dụng giao tiếp I2C để nhận dữ liệu thời gian và hiển thị lên màn hình.

-Màn hình cập nhật thời gian liên tục theo từng giây, giúp người dùng theo dõi thời gian hiện tại một cách chính xác.

Giao diện hiển thị:

-Trên màn hình OLED 0.96 inch, dòng đầu tiên hiển thị thời gian theo định dạng HH:MM:SS, ví dụ: 13:30:21.

-Dòng thứ hai hiển thị ngày và thứ, ví dụ: Tue – 08/04/2025.

-Thời gian được đồng bộ hoàn toàn tự động từ máy chủ NTP (Network Time Protocol) thông qua kết nối Wi-Fi của ESP32.

Kết luận và hướng phát triển

Việc tạo ra một chiếc đồng hồ thời gian thực sử dụng **ESP32** và màn hình **OLED 0.96** inch không chỉ là một dự án thú vị mà còn rất hữu ích trong thực tế. Với khả năng kết nối WiFi, **ESP32** có thể đồng bộ thời gian chính xác từ các máy chủ **NTP** như **Google Public NTP**, đảm bảo rằng thời gian luôn được cập nhật và chính xác. Màn hình **OLED 0.96** inch với độ phân giải cao và độ tương phản tốt giúp hiển thị thời gian một cách rõ ràng và dễ nhìn. Sự kết hợp giữa **ESP32** và màn hình **OLED** mang lại một giải pháp hiệu quả về chi phí, đồng thời cung cấp một cơ hội học tập tuyệt vời cho những ai muốn tìm hiểu về lập trình vi điều khiển và các giao thức thời gian. Dự án này có thể được mở rộng và ứng dụng trong nhiều lĩnh vực khác nhau, từ hệ thống báo thức thông minh đến các thiết bị **IoT** yêu cầu đồng bộ thời gian chính xác.

Tài liệu tham khảo

[1] Điện Thông Minh eSmart. Đồng hồ thời gian thực RTC ESP32.
https://dienthongminhesmart.com/du-an-iot/dong-ho-thoi-gian-thuc-rtc-esp32/

CÁC PHỤ LỤC

1. PHỤ LỤC A:
   Mã nguồn của chương trình Arduino:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <WiFi.h>
#include <time.h>

// Định nghĩa chiều rộng và chiều cao của màn hình OLED
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Khởi tạo đối tượng màn hình OLED
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// Định nghĩa thông tin kết nối WiFi
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// Định nghĩa UTC offset cho múi giờ Việt Nam (UTC+7)
const long gmtOffset_sec = 7 * 3600;
const int daylightOffset_sec = 0;

// Hàm lấy tên viết tắt của ngày trong tuần
const char* getDayAbbr(int wday) {
  switch (wday) {
    case 0: return "Sun";
    case 1: return "Mon";
    case 2: return "Tue";
    case 3: return "Wed";
    case 4: return "Thu";
    case 5: return "Fri";
    case 6: return "Sat";
    default: return "";
  }
}

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
```

```
  // Kết nối WiFi
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

  Serial.println("Connected to WiFi");

  // Thiết lập NTP để đồng bộ thời gian
  configTime(gmtOffset_sec, daylightOffset_sec, "time.google.com");

  // Kiểm tra và khởi động màn hình OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println("Real Time Clock");
  display.display();
  delay(2000); // Hiển thị thông điệp chào mừng trong 2 giây

  // Đồng bộ thời gian
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time");
    display.setCursor(0, 10);
    display.println("Failed to get time");
    display.display();
    while (1) { delay(1000); }
  }

  // In thời gian đồng bộ lên Serial Monitor
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
}

void loop() {
  struct tm timeinfo;
  if (getLocalTime(&timeinfo)) {
```

```
  // Xóa màn hình cũ
  display.clearDisplay();

  // Hiển thị thời gian (giờ:phút:giây)
  display.setTextSize(2);
  char timeString[10];
  sprintf(timeString, "%02d:%02d:%02d", timeinfo.tm_hour, timeinfo.tm_min,
timeinfo.tm_sec);
  Serial.println("Time: " + String(timeString));

  int16_t x1, y1;
  uint16_t width, height;
  display.getTextBounds(timeString, 0, 0, &x1, &y1, &width, &height);
  int16_t x = (SCREEN_WIDTH - width) / 2;
  display.setCursor(x, 0);
  display.print(timeString);

  // Hiển thị thứ và ngày (Thứ - ngày-tháng-năm)
  display.setTextSize(1);
  char dateString[20];
  sprintf(dateString, "%s - %02d/%02d/%04d", getDayAbbr(timeinfo.tm_wday),
timeinfo.tm_mday, timeinfo.tm_mon + 1, timeinfo.tm_year + 1900);
  display.getTextBounds(dateString, 0, 0, &x1, &y1, &width, &height);
  x = (SCREEN_WIDTH - width) / 2;
  display.setCursor(x, 40);
  display.print(dateString);

  display.display();
 }
 delay(1000); // Cập nhật mỗi giây
}
```

1. PHỤ LỤC B:

```
{
  "version": 1,
  "author": "VVH",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp32", "top": 110.3,
   "left": 91, "attrs": {} },
    {
```

```
      "type": "board-ssd1306",
      "id": "oled1",
      "top": 99.14,
      "left": 269.03,
      "attrs": { "i2cAddress": "0x3c" }
    }
  ],
  "connections": [
    [ "esp32:3V3", "oled1:VCC", "red", [ "h30", "v-30" ] ],
    [ "esp32:GND.1", "oled1:GND", "black", [ "h30", "v-20" ] ],
    [ "esp32:D21", "oled1:SDA", "green", [ "h30", "v-10" ] ],
    [ "esp32:D22", "oled1:SCL", "yellow", [ "h30", "v0" ] ]
  ],
  "dependencies": {}
}
```