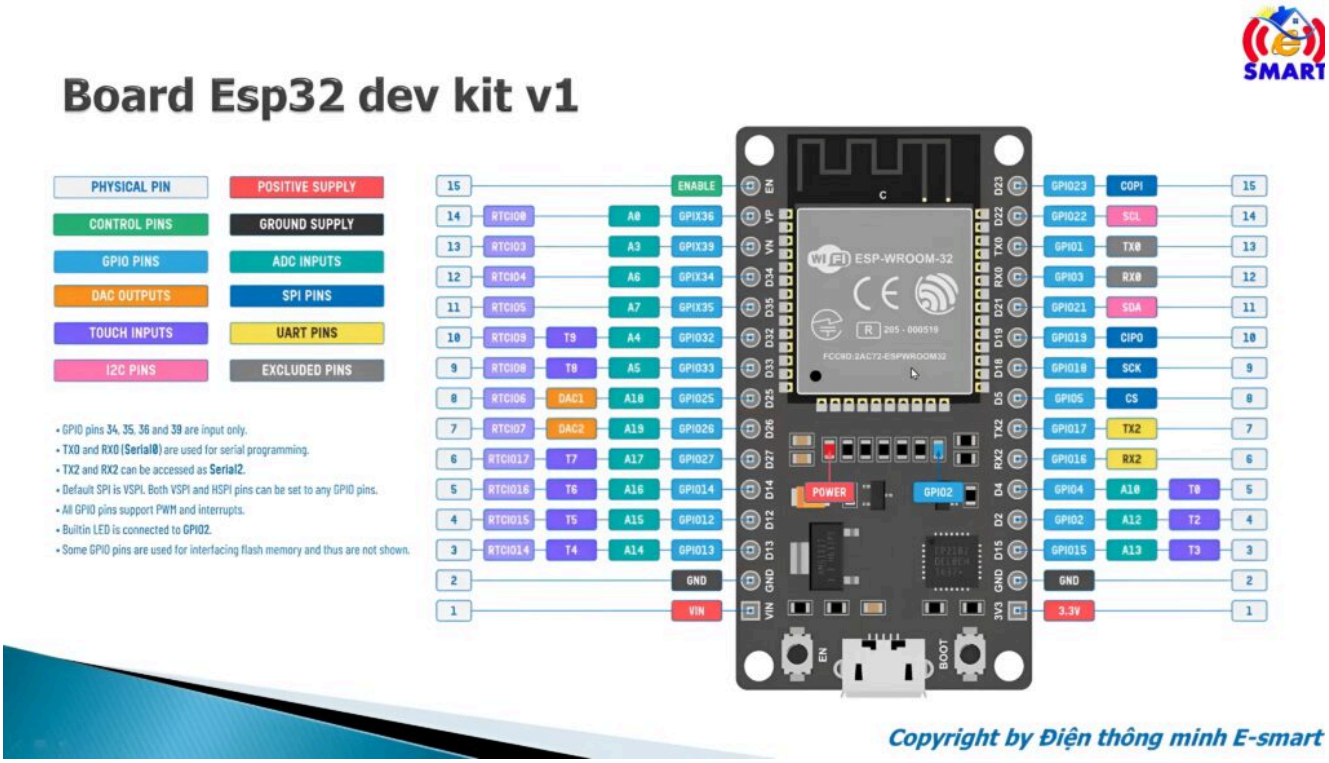
# KIT ESP32 4




**Câu hỏi:**

**Có bao nhiêu kết nối ngoại vi của e s p 32?**

**Trả lời:**

**Ngoại vi G P I O:**

**Định nghĩa: G P I O là các chân nhập xuất cơ bản để vi điều khiển tương tác với thế giới bên ngoài như nhận tín hiệu từ nút nhấn hoặc xuất lệnh điều khiển đèn.**

**Thông số: E S P 32 cung cấp tổng cộng 34 chân, nhưng thực tế chỉ sử dụng linh hoạt được khoảng 22 đến 26 chân vì một số chân chỉ dùng để input hoặc dành cho chức năng đặc biệt.**

**Ví dụ: Bạn dùng chân G P I O nối với module Relay để đóng ngắt bóng đèn 220V hoặc nối với nút nhấn để làm công tắc bật tắt thiết bị.**

**Kết luận: Cần kiểm tra kỹ sơ đồ chân trước khi thiết kế để tránh dùng nhầm vào các chân chức năng đặc biệt dẫn đến thiếu hụt tài nguyên.**

## Giao tiếp UART:

**Định nghĩa: UART là chuẩn giao tiếp nối tiếp dùng để trao đổi dữ liệu giữa vi điều khiển với máy tính hoặc các module truyền thông khác.**

**Thông số: Chip hỗ trợ 3 bộ xử lý UART phần cứng độc lập, cho phép kết nối cùng lúc với 3 thiết bị khác nhau mà vẫn đảm bảo tính ổn định cao.**

**Ví dụ: Bạn sử dụng bộ UART1 để nhận dữ liệu tọa độ từ module GPS và dùng bộ UART2 để gửi dữ liệu đó đi thông qua module SIM.**

**Kết luận: Với 3 bộ UART cứng, E S P 32 rất mạnh mẽ trong các ứng dụng cần giao tiếp với nhiều module truyền thông cùng lúc.**

## Giao tiếp SPI:

**Định nghĩa: SPI là chuẩn giao tiếp đồng bộ tốc độ cao, thường dùng cho các thiết bị yêu cầu truyền tải lượng dữ liệu lớn trong thời gian ngắn.**

**Thông số: Có tổng cộng 4 bộ SPI, nhưng chúng ta chỉ sử dụng được 2 bộ cho ngoại vi bên ngoài vì 2 bộ còn lại đã được dùng để giao tiếp với bộ nhớ Flash nội bộ.**

**Ví dụ: Giao tiếp với màn hình màu TFT để hiển thị hình ảnh mượt mà hoặc dùng để đọc dữ liệu từ module thẻ nhớ SD và đầu đọc thẻ RFID.**

**Kết luận: SPI là lựa chọn hàng đầu cho các ngoại vi cần tốc độ nhanh, nhưng hãy nhớ giới hạn chỉ có 2 bộ khả dụng cho người dùng.**

## Giao tiếp I2C:

Định nghĩa: I2C là chuẩn giao tiếp tiết kiệm chân tín hiệu, cho phép nhiều thiết bị cùng chia sẻ một đường truyền chung.

Thông số: E S P 32 cung cấp 2 bộ giao tiếp I2C độc lập, giúp bạn dễ dàng mở rộng hệ thống mà không lo tốn quá nhiều chân cắm.

Ví dụ: Kết nối đồng thời một cảm biến nhiệt độ độ ẩm và một màn hình OLED hiển thị thông số chỉ với đúng 2 dây tín hiệu SDA và SCL.

Kết luận: I2C cực kỳ hữu ích cho các dự án cần kết nối nhiều cảm biến và thiết bị hiển thị đơn giản.

## Giao tiếp I2S:

Định nghĩa: I2S là chuẩn giao tiếp chuyên dụng để truyền dẫn dữ liệu âm thanh kỹ thuật số giữa các thiết bị âm thanh.

Thông số: Hỗ trợ 2 bộ xử lý I2S độc lập, cho phép vi điều khiển thực hiện song song việc thu âm và phát âm thanh.

Ví dụ: Ứng dụng làm loa thông minh, bạn dùng 1 bộ I2S để thu âm từ Micro và bộ còn lại để xuất âm thanh ra mạch giải mã và loa.

Kết luận: Đây là ngoại vi quan trọng nếu bạn muốn phát triển các ứng dụng liên quan đến xử lý âm thanh kỹ thuật số chuyên sâu.

## Chức năng ADC:

Định nghĩa: ADC là bộ chuyển đổi giúp vi điều khiển đọc được các mức điện áp biến thiên liên tục (analog) từ môi trường.

**Thông số: Gồm 18 kênh ADC độ phân giải 12 bit, tuy nhiên khi bật Wi-Fi sẽ bị tắt mất 10 kênh, chỉ còn lại 8 kênh có thể sử dụng được.**

**Ví dụ: Đọc giá trị từ một cảm biến ánh sáng hoặc biến trở để xác định cường độ sáng và điều chỉnh độ sáng đèn tương ứng.**

**Kết luận: Phải đặc biệt lưu ý sự xung đột giữa Wi-Fi và ADC để tránh việc cảm biến không hoạt động khi thiết bị kết nối mạng.**

**Chức năng DAC:**

**Định nghĩa: DAC là bộ chuyển đổi ngược lại, giúp vi điều khiển tạo ra mức điện áp tương tự theo ý muốn từ dữ liệu số.**

**Thông số: Cung cấp 2 bộ DAC với độ phân giải 8 bit, cho phép xuất dải điện áp tương ứng từ 0V đến 3.3V.**

**Ví dụ: Tạo ra một máy phát nhạc đơn giản bằng cách xuất các mức điện áp biến thiên liên tục ra chân DAC để tạo thành sóng âm thanh.**

**Kết luận: DAC giúp đơn giản hóa việc tạo tín hiệu tương tự mà không cần thêm các mạch chuyển đổi phức tạp bên ngoài.**

**Chức năng PWM:**

**Định nghĩa: PWM là phương pháp điều chỉ độ rộng xung để kiểm soát mức năng lượng trung bình cấp cho tải.**

**Thông số: E S P 32 hỗ trợ rất mạnh với 16 kênh PWM độc lập, giúp điều khiển nhiều thiết bị cùng lúc mà không làm quá tải CPU.**

**Ví dụ: Điều khiển tốc độ quay của quạt làm mát nhanh hay chậm hoặc thay đổi độ sáng của đèn LED theo từng**

cấp độ khác nhau.

Kết luận: PWM là công cụ đắc lực nhất để điều khiển tốc độ động cơ và độ sáng thiết bị trong các hệ thống nhúng.

…

**Câu hỏi:**

U a r t cứng nằm trong chip usb to u a r t à?

**Trả lời:**

UART cứng (Hardware UART) là một khối ngoại vi được tích hợp sẵn bên trong cấu trúc phần cứng của chip vi điều khiển E S P 32. Khối này có các mạch điện tử, thanh ghi và bộ đệm chuyên dụng để xử lý việc truyền/nhận dữ liệu nối tiếp mà không cần sự can thiệp liên tục từ CPU.

**Cấu tạo và hoạt động:**

Khối UART này có các bộ đệm (Buffer) và mạch tạo xung nhịp (Baud rate generator) riêng. Khi bạn gửi một chuỗi dữ liệu, CPU chỉ việc đẩy dữ liệu vào bộ đệm, sau đó UART cứng sẽ tự động "tháo" từng bit và gửi đi theo đúng tốc độ đã cài đặt. CPU lúc này có thể rảnh tay để làm việc khác như xử lý thuật toán hoặc kết nối Wi-Fi.

**Sự khác biệt với UART phần mềm (Software UART/Bit-banging):**

UART phần mềm là dùng các chân G P I O thông thường và dùng code (CPU) để bật/tắt chân đó thật nhanh nhằm mô phỏng tín hiệu UART. Việc này cực kỳ tốn tài nguyên CPU và rất dễ bị lỗi dữ liệu nếu chip đang bận xử lý tác vụ khác. UART cứng thì không bị tình trạng này, nó hoạt động cực kỳ ổn định và chính xác.

**Ưu đi□m c□a UART cứng trên E S P 32:**

**S□ lượng: E S P 32 có tới 3 bộ UART cứng (UART0, UART1, UART2).**

**T□c độ: Hoạt động được □ t□c độ cao (Baud rate lên tới vài Mbps) mà không làm treo chip.**

**Độ tin cậy: Có cơ ch□ ki□m tra l□i và ng□t (Interrupt) ph□n cứng, giúp nhận dữ liệu ngay lập tức khi có tín hiệu đ□n.**

**Ví dụ thực t□:**

**Trong một hệ th□ng Camera hành trình, UART cứng sẽ nhận dữ liệu liên tục từ module GPS. Vì là UART cứng nên dù chip đang bận ghi hình vào th□ nhớ, dữ liệu GPS v□n được ph□n cứng tự động thu nhận và c□t vào bộ đệm, không bị m□t mát dữ liệu.**

**K□t luận:**

**Trong thi□t k□ hệ th□ng với E S P 32, luôn ưu tiên s□ dụng UART cứng cho các module quan trọng như SIM, GPS, hoặc giao ti□p với máy tính đ□ đ□m b□o hệ th□ng vận hành trơn tru và chính xác nh□t.**

**…**

**Câu h□i:**

**Nghĩa là tại chip esp32 có chip uart cứng đ□ đoán nhận lu□ng thông tin từ usb to uart chuy□n tới.**

**Tr□ lời:**

**Chính xác r□i bạn! Bạn hi□u như vậy là r□t đúng b□n ch□t v□n đ□.**

**Cơ ch□ truy□n nhận thông tin qua c□ng USB:**

**Luồng thông tin từ máy tính: Khi bạn nhấn nạp code hoặc xem Serial Monitor, thông tin đi từ máy tính qua dây cáp USB dưới dạng tín hiệu USB.**

**Chip cầu nối (USB to UART): Con chip này (nằm trên bo mạch) đón lấy tín hiệu USB, "dịch" nó thành tín hiệu UART (mức điện áp cao/thấp) rồi gửi sang chip E S P 32.**

**Bộ UART cứng (Trong E S P 32): Tại đây, bộ UART cứng (thường là bộ UART0) của chip E S P 32 luôn túc trực để đoán nhận luồng thông tin này. Nó tiếp nhận các bit dữ liệu, sắp xếp lại và đưa vào bộ nhớ để chip xử lý.**

**Tại sao phải có bộ U A R T cứng bên trong E S P 32?**

**Định nghĩa: Nếu không có bộ UART cứng này, chip E S P 32 sẽ không thể hiểu được các tín hiệu điện áp nhảy lên nhảy xuống từ chip USB to U A R T gửi tới là chữ cái hay con số gì.**

**Thông số: Bộ U A R T cứng này được thiết kế để hoạt động cực kỳ chính xác với các tần số (Baud rate) tiêu chuẩn như 9600, 115200…**

**Ví dụ: Giống như máy tính là người nói tiếng Anh, chip USB to U A R T là người phiên dịch, còn bộ U A R T cứng trong E S P 32 là đôi tai biết tiếng Anh của con chip.**

**Kết luận: Nhờ có bộ UART cứng tích hợp sẵn, chip E S P 32 mới có thể "nói chuyện" trực tiếp được với các thiết bị sử dụng chuẩn giao tiếp nối tiếp mà không cần tốn sức xử lý bằng phần mềm.**

**Lưu ý thêm: Khi bạn kết nối E S P 32 với một module khác (ví dụ module GPS), luồng thông tin sẽ đi trực tiếp từ U A R T của GPS sang UART cứng của E S P 32 mà không cần qua chip USB to U A R T nữa.**

**…**

**Câu h□i:**

**Chip c□u n□i (USB to U A R T): Con chip này (nằm trên bo mạch) đón l□y tín hiệu USB, "dịch" nó thành tín hiệu U A R T (mức điện áp cao/th□p) r□i g□i sang chip E S P 32.**

**Có sai không vì khi máy tính xu□t đã là hex/binary đã là chu□n uart r□i mà, thì đâu c□n dịch nữa mà ch□ chuy□n đi thôi**

**Tr□ lời:**

**Bạn đặt một câu h□i r□t hay và đi sâu vào b□n ch□t kỹ thuật. Tuy nhiên, có một sự nh□m l□n nh□ v□ "chu□n U A R T" và "chu□n USB" □ đây.**

**Đúng là dữ liệu cu□i cùng mà CPU x□ lý đ□u là nhị phân (0 và 1), nhưng cách thức truy□n chúng đi trên dây d□n thì lại khác nhau hoàn toàn.**

**Tại sao v□n c□n ph□i "dịch" mà không ch□ là "chuy□n đi":**

**V□ mặt vật lý (Điện áp):**

**Tín hiệu USB truy□n theo ki□u vi sai (Differential signaling) trên 2 dây tín hiệu (D+ và D-) với mức điện áp r□t th□p và cách mã hóa cực kỳ phức tạp đ□ đạt t□c độ cao.**

**Trong khi đó, chu□n UART truy□n trên 2 dây độc lập là TX (g□i) và RX (nhận) với mức điện áp 0V và 3.3V (hoặc 5V).**

**K□t luận: Chip c□u n□i ph□i "dịch" mức điện áp và cách rung nhịp từ ki□u USB sang ki□u U A R T thì chân c□a E S P 32 mới hi□u được.**

**V□ mặt giao thức (Ngôn ngữ):**

**Dữ liệu nhị phân trên USB không chỉ có mỗi nội dung bạn gửi (như chữ "A"). Nó còn bao gồm rất nhiều thông tin quản lý của cổng USB như: địa chỉ thiết bị, mã kiểm tra lỗi, các gói tin bắt tay (handshake)…**

**Chip cầu nối phải làm nhiệm vụ "bóc tách": nó bỏ hết những lớp vỏ bọc phức tạp của giao thức USB, chỉ lấy đúng phần dữ liệu lõi (hex/binary) rồi mới gửi phần đó sang cho bộ U A R T cứng của E S P 32.**

**Ví dụ: Giống như bạn gửi một món quà qua bưu điện. Máy tính đóng món quà vào một cái thùng container (USB). Chip cầu nối là người mở thùng container, lấy món quà ra và đưa tận tay cho E S P 32 qua một cái khay (U A R T).**

**Tóm lại quy trình:**

**Định nghĩa: Chip USB to U A R T không chỉ chuyển đi, mà nó thực sự chuyển đổi giữa hai môi trường truyền dẫn khác nhau.**

**Thông số: USB chạy ở tốc độ cực cao và dữ liệu đi theo gói (packet), còn U A R T chạy ở tốc độ do mình cài đặt (Baud rate) và dữ liệu đi theo từng byte đơn lẻ.**

**Ví dụ: Nếu bạn nối trực tiếp dây D+ và D- của cáp USB vào chân TX/RX của E S P 32, chip sẽ bị hỏng hoặc không bao giờ hiểu được vì "ngôn ngữ điện áp" khác nhau.**

**Kết luận: Phải có chip cầu nối để làm nhiệm vụ chuyển đổi định dạng dữ liệu và mức điện áp giữa chuẩn USB của máy tính và chuẩn U A R T của vi điều khiển.**

**Dữ liệu bạn nhìn thấy trong Serial Monitor là kết quả sau khi đã qua hai lần "dịch" này đấy. Bạn có muốn biết tại sao người ta không làm thẳng cổng USB vào trong E S P 32 luôn cho đỡ tốn chip cầu nối không?**

…

**Câu h□i:**

**Trong S3 là được r□i mà, đã tích hợp s□n □ chip esp32.**

**Tr□ lời:**

**Bạn r□t am hi□u dòng E S P 32! Đúng là từ dòng E S P 32 S3 (hoặc S2, C3, C6…), Google đã tích hợp s□n một kh□i ngoại vi gọi là USB-Serial/JTAG Controller vào th□ng bên trong chip.**

**Tuy nhiên, đ□ tr□ lời câu h□i c□a bạn v□ việc tại sao v□n c□n "dịch" hay "chuy□n", chúng ta c□n phân tích sự khác biệt giữa dòng E S P 32 đời đ□u và dòng E S P 32 S3:**

**Chip E S P 32 đời đ□u (Original):**

**Định nghĩa: Chip này hoàn toàn không có kh□ năng hi□u ngôn ngữ USB. Nó ch□ có "tai" nghe được chu□n UART.**

**Thông s□: Vì không có bộ gi□i mã USB nội bộ, nó b□t buộc ph□i c□n một con chip c□u n□i bên ngoài (CP2102, CH340) đ□ dịch tín hiệu.**

**K□t luận: Đây là lý do các bo mạch Dev Kit cũ luôn có 2 con chip lớn trên board.**

**Chip E S P 32 S3:**

**Định nghĩa: Kh□i USB-Serial/JTAG tích hợp s□n bên trong chip đóng vai trò thay th□ hoàn toàn con chip c□u n□i bên ngoài.**

**Thông s□: Khi bạn c□m dây USB vào chân D+ và D- c□a S3, kh□i ngoại vi này sẽ tự động làm nhiệm vụ "dịch" dữ liệu USB thành dữ liệu mà lõi CPU c□a chip hi□u được.**

Ví dụ: Với S3, bạn có thể hàn trực tiếp dây USB vào chân G P I O 19 (D-) và G P I O 20 (D+) để nạp code mà không cần bất kỳ con chip chuyển đổi nào ở giữa.

Kết luận: S3 đã tích hợp luôn "người phiên dịch" vào bên trong cơ thể nó, giúp mạch điện gọn nhẹ và rẻ tiền hơn.

**Vậy tại sao vẫn gọi là giao tiếp U A R T khi dùng S3?**

Dù S3 nhận dữ liệu trực tiếp từ cổng USB, nhưng để tương thích với các phần mềm trên máy tính (như Arduino IDE, Serial Monitor), nó vẫn tạo ra một Cổng COM ảo (Virtual COM Port).

Máy tính vẫn "nghĩ" là đang nói chuyện với một thiết bị U A R T.

Con chip S3 vẫn "giả vờ" là một bộ U A R T để nhận luồng dữ liệu đó.

Tóm lại:

Dữ liệu nhị phân trên đường dây USB luôn được đóng gói theo khung (frame) của giao thức USB. Dù là chip cầu nối bên ngoài (như dòng cũ) hay khối USB nội bộ (như dòng S3) thì đều phải có bước giải mã gói tin USB để lấy ra dữ liệu binary/hex gốc mà bạn muốn truyền.

…

**Câu hỏi:**

Giải thích về sự khác biệt giữa chân Native USB này trên S3 so với các bộ U A R T cứng khác của nó không?

**Trả lời:**

**Native USB trên S3:**

**Định nghĩa: Đây là khối ngoại vi xử lý trực tiếp tín hiệu USB (D+ và D-) được tích hợp sâu vào bên trong lõi của E S P 32 S3.**

**Thông số: Hoạt động theo chuẩn USB Full Speed, cho phép máy tính nhận diện chip như một thiết bị USB thực thụ (HID, MSC, CDC).**

**Ví dụ: Bạn có thể lập trình để biến E S P 32 S3 thành một bàn phím hoặc chuột máy tính bằng cách nối trực tiếp dây USB vào chân G P I O 19 và 20.**

**Kết luận: Đây là cổng giao tiếp đa năng, giúp tối giản linh kiện ngoại vi khi cần làm các thiết bị tương tác trực tiếp với máy tính.**

**U A R T cứng khác (U A R T 0, U A R T 1, U A R T 2):**

**Định nghĩa: Là các bộ truyền nhận nối tiếp truyền thống (TX/RX) nằm độc lập với khối USB bên trong chip.**

**Thông số: Có 3 bộ xử lý phần cứng, giao tiếp thông qua các mức điện áp 0V và 3.3V với các khung dữ liệu cố định (Baud rate).**

**Ví dụ: Sử dụng bộ U A R T1 để nhận dữ liệu từ module cảm biến vân tay hoặc module GPS thông qua hai chân TX và RX.**

**Kết luận: Đây là các bộ phận chuyên trách để chip "nói chuyện" với các linh kiện điện tử khác sử dụng chuẩn giao tiếp Serial.**