**GIỚI THIỆU**

**ESP32 – FreeRTOS**

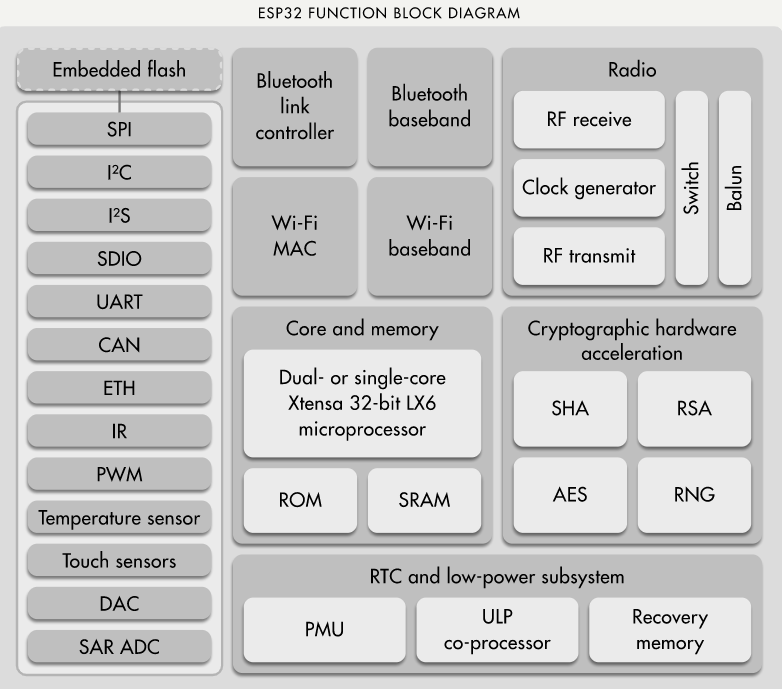
1. **Lý thuyết về ESP32**

**1.1 Giới thiệu ESP32**

- ESP32 là một series các vi điều khiển trên một vi mạch giá rẻ, năng lượng thấp có tích hợp WiFi và dual-mode Bluetooth (tạm dịch: Bluetooth chế độ kép). Dòng ESP32 sử dụng bộ vi xử lý Tensilica Xtensa LX6 có hai biến thể lõi kép và lõi đơn, và bao gồm các công tắc antenna tích hợp, RF balun, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng.

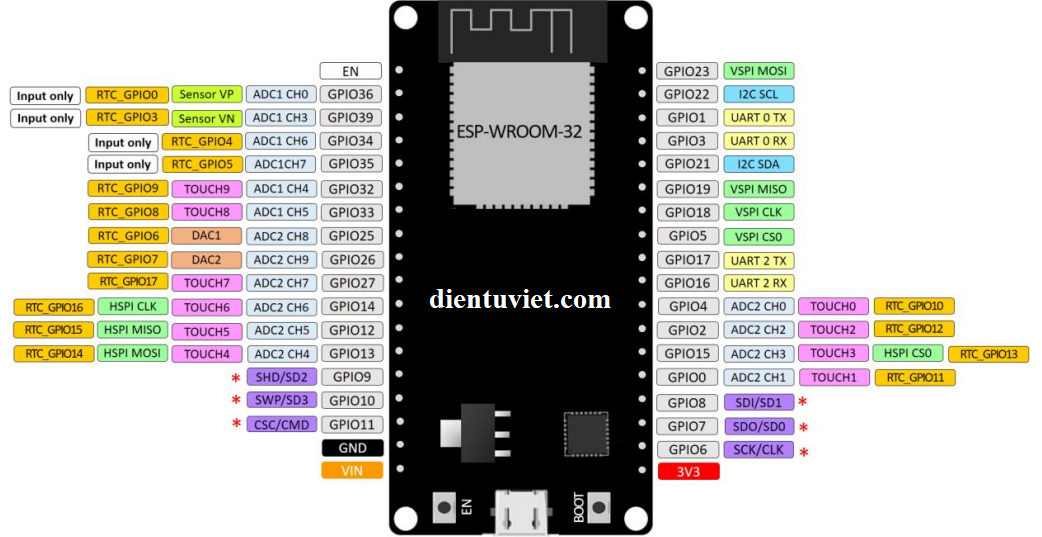
**ESP32** được chế tạo và phát triển bởi Espressif Systems, một công ty Trung Quốc có trụ sở tại Thượng Hải, và được sản xuất bởi [TSMC](https://vi.wikipedia.org/wiki/TSMC) bằng cách sử dụng công nghệ 40 nm. ESP32 là sản phẩm kế thừa từ vi điều khiển ESP8266.

* 1. **Cấu hình của ESP32**



**CPU**

* CPU: được đặt tên là “PRO\_CPU” và “APP\_CPU” Xtensa Dual-Core LX16. Khi chúng ta dung FreeRTOS sẽ ứng với Core 0 và Core 1
* CPU: Xtensa Dual-Core LX6 microprocessor.
* Chạy hệ 32 bit
* Tốc độ xử lý từ 160 MHz đến 240 MHz
* ROM: 448 Kb
* Tốc độ xung nhịp từ 40 Mhz ÷ 80 Mhz (có thể tùy chỉnh khi lập trình)
* RAM: 520 Kb SRAM liền chip. Trong đó 8 Kb RAM RTC tốc độ cao – 8 Kb RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep).
  1. **Sơ đồ chân của ESP32**



▪ Có 18 kênh – bộ chuyển đổi ADC – Analog-to-Digital

▪ 3xSPI, 3xUART, 2xI2C,

▪ 16 kênh đầu ra PWM,

▪ 2 Bộ chuyển đổi DAC- Digital-to-Analog

▪ 2 x I2S, 10 GPIO cảm biến điện dung.

▪ Các tính năng của ADC và DAC sẽ được gán trực tiếp vào các chân cố định. Tuy nhiên, người dùng có thể lựa chọn các chân như SPI, PWM, UART, I2C, v.v. Chúng ta có thể dễ dàng làm điều này bằng việc khai báo chân trong code. Việc này được phép thực hiện nhờ tính năng ghép kênh của chip ESP32.

VD: Các chân từ 34 39 là các chân Input only pins vì thế người dùng không thể thiết lập chúng là chân Output.

* 1. **Môi trường lập trình ESP32**

ESP32 có thể được lập trình trong các môi trường lập trình khác nhau như là:

1. Arduno IDE
2. Espressif IDF
3. Micropython
4. JavaScript
5. LUA
6. **Lý thuyết về FreeRTOS**

**2.1 Khái niệm FreeRTOS**

- FreeRTOS là một hệ điều hành nhúng thời gian thực (real time operating system ) mã nguồn mỡ ,được phát triễn bởi Real Time Engineers Ltd,sáng lập và sở hữu bởi Richard barry

Phù hợp cho nhiều hệ nhúng nhỏ gọn vì nó chỉ triển khai rất ít các chức năng như :cơ chế quản lí bộ nhớ và các tác vụ cơ bản ,các hàm API quan trọng cho cơ chế đồng bộ.

Nó không cung cấp sẵn các giao tiếp mạng, drivers,hay hệ thống quả lý tệp (file system) như những hệ điều hành khác.

Tuy vậy FreeRTOS có nhiều ưu điểm :

* Hổ trợ nhiều kiến trúc vi điều khiển khác nhau (PIC ,VAR,MSP430,ARM……)
* Kích thước nhỏ gọn
* Phát triển với nhiều trình biên dịch C khác nhau (GCC,OPenWatcom,Keil,IAR,Eclipse,…..)
* Cho phép không giới hạn các tác vụ chạy đồng thời ,không hạn chế quyền ưu tiên thực thi ,khả năng khai thác phần cứng .
* Ngoài ra nó cũng cho phép triễn khai các cơ chế điều độ giữa các tiến trình như : hàng đợi(queue),cờ đếm(counting semaphone),loại trừ lẫn nhau (mutex)

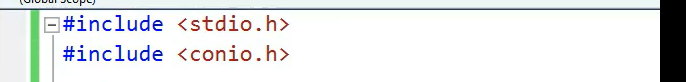
**2.2 Đặc điểm của FreeRTOS**

Các chức năng được cung cấp trong FreeRTOS

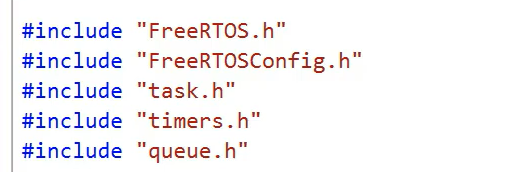
FreeRTOS là lõi của hệ điều hành thời gian thực được thiết kế cho các hệ thống nhúng nhỏ, với các chức năng chính sau :

* Lõi FreeRTOS hỗ trợ cả preemptive, cooperative và các lựa chọn cấu hình lai giữa hai phần.
* Được thiết kế nhỏ, đơn giản và dễ sử dụng. Cấu trúc mã nguồn rất linh động được viết bằng ngôn ngữ C.
* Có lựa chọn nhận biết tràn ngăn xếp.
* Không giới hạn số task có thể tạo ra, phụ thuộc vào tài nguyên của chip. Không giới hạn số mức ưu tiên được sử dụng. Không giới hạn số task cùng một mức ưu tiên.
* Hỗ trợ truyền thông và đồng bộ giữa các task hoặc giữa task và ngắt: queues, binary semaphores, counting semaphores and recursive mutexes.Mutexes v ới ưu tiên kế thừa. Các công cụ phát triển miễn phí, port cho Cortex -M3, ARM7, PIC,MSP430, H8/S, AMD, AVR, x86 và 8051.Miễn phí mã nguồn phần mềm nhúng. Miễn phí trong ứng dụng thương mại.
  1. **Cấu trúc chương trình FreeRTOS**
* Cấu trúc chung 1 chương trình freeRTOS :

**Header file** : chứa các file.h trong hệ điều hành FreeRTOS

Gồm các thư viện có sẵn ****

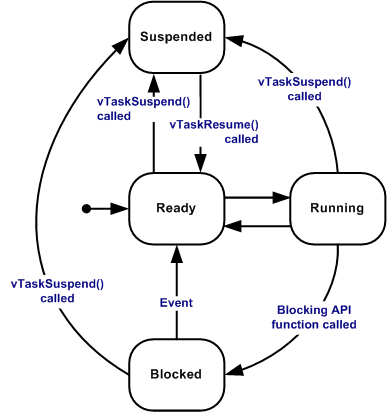
Và các thư viện được thêm vào trong hệ điều hành FreeRTOS

****

**Source file :** Chứa các source file,các hàm của chương trình chính , thường có đuôi .C.vì FreeRTOS được viết bằng chương trình C

* 1. **Các trạng thái (states) của task**

Có 4 trạng thái: Running, Ready, Blocked và Suspended.



Running: là trạng thái task đang được MCU thực thi thực sự, vì thế trong một thời điểm chỉ có duy nhất một task ở trạng thái running.

Ready: là trạng thái task đã sẵn sàng để thực thi (không bị blocked hoặc suspended) nhưng đang không được MCU thực thi bởi vì MCU đang thực thi một task khác.

Blocked: task đang bị blocked nếu nó đang đợi sự kiện (event) bên ngoài hoặc sự kiện thời gian. Ví dụ: khi task gọi hàm vTaskDelay() thì nó sẽ bị blocked cho đến khi hết thời gian delay (sự kiện thời gian). Hoặc task có thể bị blocked để đợi một hàng đợi (queue), semaphore, notification… nào đó. Thông thường, task bị blocked trong khoảng thòi gian quá hạn timeout cho trước, vì thế task sẽ luôn được unblocked (nếu có sự kiện bên ngoài) hoặc timeout nếu hết thời gian chờ.

Suspended: Cũng giống như trạng thái blocked nhưng không có thời gian timeout, vì vậy chỉ có thể enter hoặc exit khỏi trạng thái suspended bởi hàm gọi từ bên ngoài tương ứng là vTaskSuspend() và xTaskResume().