Guía de Referencia: El Microcontrolador ESP32 para Principiantes

1. Introducción al Mundo de los Microcontroladores

En el ecosistema de la electrónica moderna, la capacidad de dotar de "inteligencia" a los objetos cotidianos ha transformado nuestra interacción con el entorno. Los microcontroladores actúan como el cerebro estratégico detrás de esta revolución, gestionando desde la lógica de un electrodoméstico inteligente hasta los sistemas críticos de la automatización industrial. Es fundamental entender que estos componentes no son simples ejecutores de código; son puentes que traducen el software abstracto en acciones físicas tangibles.

Técnicamente, el ESP32 se define como un **Sistema en un Chip (SoC)**. A diferencia de un microprocesador convencional, un SoC integra en un único trozo de silicio el procesador, la memoria y los controladores de entrada/salida. Esta integración permite que el dispositivo sea extremadamente económico y de bajo consumo, diseñado específicamente para controlar tareas dedicadas con una eficiencia que las computadoras de propósito general no pueden igualar.

¿Y qué impacto tiene esto? La evolución desde chips simples hacia sistemas integrados de alto rendimiento como el ESP32 ha democratizado el acceso a la creación tecnológica. Hoy, un estudiante o un entusiasta puede desarrollar sistemas de conectividad que hace una década requerían infraestructuras de ingeniería masivas. El ESP32 destaca en este universo por ofrecer capacidades inalámbricas y de procesamiento que antes eran inalcanzables a este nivel de costo, abriendo la puerta a una arquitectura interna sofisticada.

---

2. ¿Qué es el ESP32 y por qué es Especial?

El ESP32 no es una evolución incremental; es un salto disruptivo. Creado por **Espressif Systems** como el sucesor potenciado del ESP8266, este chip está fabricado con tecnología de **40 nanómetros (nm)**, lo que permite una alta densidad de componentes en un espacio mínimo. Su posición actual como herramienta líder se debe a su excepcional equilibrio entre potencia de procesamiento y versatilidad inalámbrica.

Lo que realmente lo define es su conectividad nativa: integra **Wi-Fi 802.11 b/g/n** y **Bluetooth de modo dual**, cumpliendo con los estándares **v4.2 BR/EDR (Classic)** y **BLE (Low Energy)**. Esta dualidad es vital: mientras el Wi-Fi permite la conexión a infraestructuras de red globales, el Bluetooth Low Energy facilita la comunicación directa con dispositivos móviles con un gasto energético mínimo.

¿Y qué impacto tiene esto? Al incluir Wi-Fi y Bluetooth en un mismo chip, se elimina la necesidad de periféricos externos costosos y complejos de diseñar. Para el creador, esto reduce drásticamente la barrera de entrada para proyectos de **Internet de las Cosas (IoT)**. Podemos

construir desde sensores ambientales que reportan a la nube hasta cerraduras inteligentes con control por proximidad, todo desde una única plataforma compacta. Sin embargo, para dominar esta herramienta, primero debemos analizar la "anatomía" técnica que permite tal rendimiento.

---

3. Anatomía y Componentes del ESP32

Para un desarrollador, conocer el interior del chip es la diferencia entre escribir un código funcional y diseñar un sistema optimizado. El ESP32 es una bestia de procesamiento en miniatura, diseñada para manejar flujos de datos intensos sin sacrificar la autonomía de la batería.

Sus componentes principales incluyen:

- **Procesadores:** El núcleo del sistema es el microprocesador **Xtensa® LX6 de 32 bits**. Dependiendo del modelo (como el WROOM-32), puede operar a 160 o 240 MHz, alcanzando un rendimiento impresionante de hasta **600 DMIPS** (millones de instrucciones por segundo).
- **Memoria:** Dispone de **520 KiB de SRAM** interna para datos y aplicaciones. Además, módulos populares como el **Lolin32** suelen incorporar **4MB de memoria Flash** para el almacenamiento del programa.
- **Módulos de Radio:** Más allá de la antena, incluye componentes críticos como el **RF balun**, interruptores de antena, un amplificador de potencia y un **amplificador de recepción de bajo ruido (LNA)** que garantiza estabilidad incluso con señales débiles.
- **Coprocesador ULP:** El coprocesador de **Ultra Bajo Consumo** es una joya pedagógica: permite realizar mediciones analógicas y monitorear sensores mientras los núcleos principales están en sueño profundo, consumiendo apenas microamperios.

¿Y qué impacto tiene esto? La integración de un regulador interno y módulos de gestión de energía avanzados permite que el dispositivo sea alimentado por baterías de litio durante meses. El ULP puede "despertar" al procesador principal solo cuando ocurre un evento crítico, optimizando la durabilidad del proyecto. No obstante, este poder de procesamiento sería inútil si no tuviera un camino físico para interactuar con el exterior: sus pines.

---

4. Guía Detallada del PINOUT (Pines de Entrada y Salida)

Los pines son la interfaz física que permite al microcontrolador "sentir" el entorno mediante sensores y "actuar" a través de actuadores. Es vital comprender que el ESP32 utiliza una lógica de voltaje de **3.3V**; conectarlo directamente a 5V en sus pines de datos puede causar daños permanentes.

A continuación, presento una referencia rápida de los pines más importantes basada en el estándar del módulo WROOM-32:

Función	Pines Específicos (GPIO)	Notas Técnicas
Alimentación	3.3V, Vin (5V), GND	Vin acepta 5V regulados por el chip.
I2C (Default)	SDA (GPIO21), SCL (GPIO22)	Estándar para pantallas y sensores.
ADC (Entradas)	GPIO32 a GPIO39, entre otros	**12-bit SAR ADC** (0 a 4095).
DAC (Salidas)	GPIO25 (DAC1), GPIO26 (DAC2)	**8-bit DAC** (convierte digital a voltajes).
Touch	GPIO4 (T0), GPIO15 (T3), GPIO32 (T9)	10 sensores táctiles capacitivos integrados.

- **Nota de Experto (Sensores Internos):** El ESP32 incluye un **sensor de efecto Hall** integrado para detectar campos magnéticos. Es importante notar que este es un sensor interno y no requiere un pin físico para su lectura (se usa la función `hallRead()`).
- **Pro-Tip sobre Multiplexación:** Muchos pines son multipropósito. Un mismo pin puede ser configurado como digital, analógico o táctil. Sin embargo, no puede cumplir todas las funciones simultáneamente; el diseño debe ser selectivo.

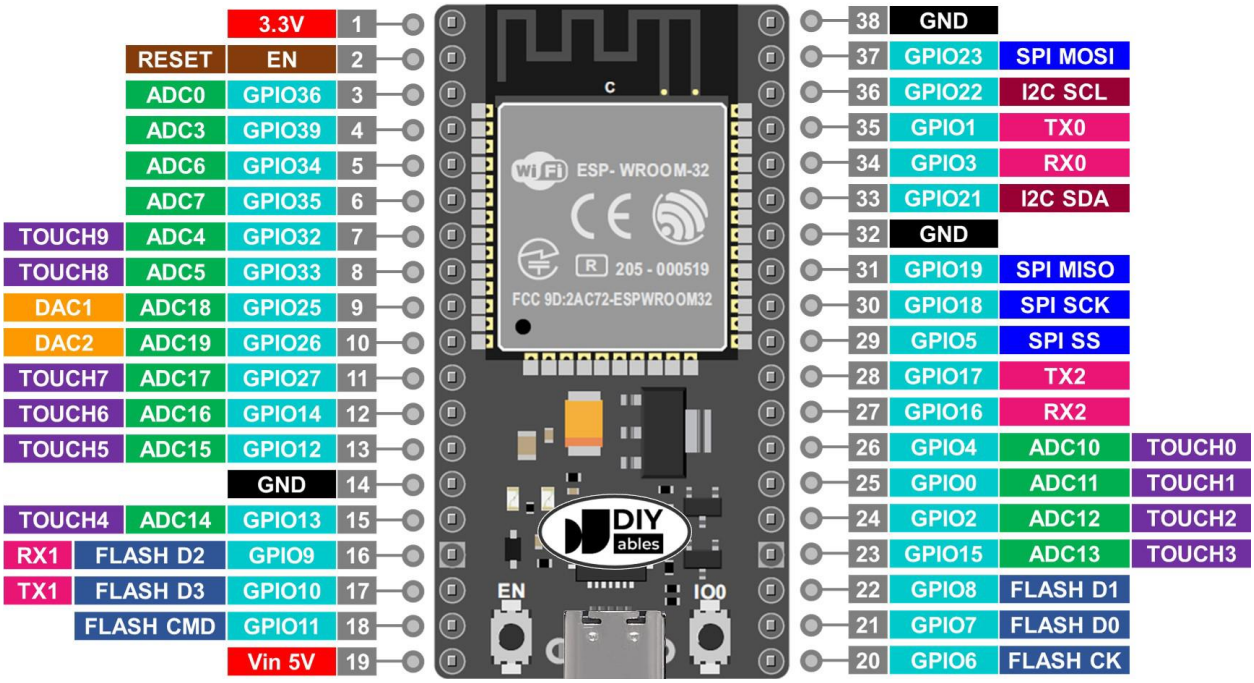

Imagen 1: Diagrama de pines (pinout) para una placa de desarrollo ESP32-WROOM-32 de 38 pines, específicamente de la marca DIYables.

Fuente: https://europe1.discourse-cdn.com/arduino/optimized/4X/c/7/2/c720a5af74f8e589f3c79258ed905e38aff9f0e0_2_690x392.jpeg

¿Y qué impacto tiene esto? La versatilidad de los **18 canales ADC de 12 bits** permite una precisión de lectura de sensores muy superior a la de placas clásicas como el Arduino Uno (que

es de 10 bits). Esto, sumado a la capacidad de convertir datos digitales en voltajes reales mediante los DACs, convierte al ESP32 en una herramienta profesional de bajo costo. Esta gestión masiva de pines es posible gracias a que el chip posee dos "cerebros" trabajando en paralelo.

---

5. El Concepto de Doble Núcleo (Dual Core)

A diferencia de los microcontroladores más simples que ejecutan tareas de forma lineal (una tras otra), el ESP32 posee una arquitectura de doble núcleo simétrico que le permite realizar multitarea real.

- **Arquitectura:** Posee dos núcleos idénticos: el **Core 0 (Protocol CPU)**, usualmente encargado del Wi-Fi y Bluetooth, y el **Core 1 (Application CPU)**, donde se ejecuta el código del usuario.
- **Diferencias en Programación:**
    - **Uso Estándar (Arduino IDE):** El código corre por defecto en el Core 1. Para aprovechar ambos núcleos, se utilizan funciones de FreeRTOS como `xTaskCreatePinnedToCore`.
    - **Uso Industrial:** En sistemas como los PLCs de *Industrial Shields*, se ha simplificado esta lógica permitiendo el uso de una función secundaria llamada `loop1()` para tareas del segundo núcleo.
- **Sincronización y Semáforos:** Cuando ambos núcleos comparten una variable global, pueden ocurrir conflictos. Para evitarlo, se usan **semáforos** mediante tres funciones clave:

1. `take()`: Bloquea el recurso para uso exclusivo de un núcleo.
2. `give()`: Libera el recurso para el otro núcleo.
3. `isTaken()`: Verifica si el recurso está ocupado.

¿Y qué impacto tiene esto? Esta arquitectura garantiza la fiabilidad del sistema. En un proyecto de IoT, el Core 0 puede mantener la conexión Wi-Fi estable y responder a peticiones de red sin que el Core 1 tenga que detener sus cálculos matemáticos o la lectura crítica de sensores. Esta separación de responsabilidades evita bloqueos (freezes) y asegura que el dispositivo nunca pierda conectividad, una característica esencial para aplicaciones profesionales.

---

6. Bibliografía

- Boot & Work Corp. S.L. (20 de abril de 2021). *Cómo utilizar el PLC industrial ESP32 de doble núcleo*. Recuperado de Industrial Shields Blog.
- Espressif Systems. (2023). *ESP32 Technical Reference Manual (Version 5.1)*. Espressif Systems.
- Hendry, I. (2023). *ESP32 Development using the Arduino IDE*. Editorial independiente.