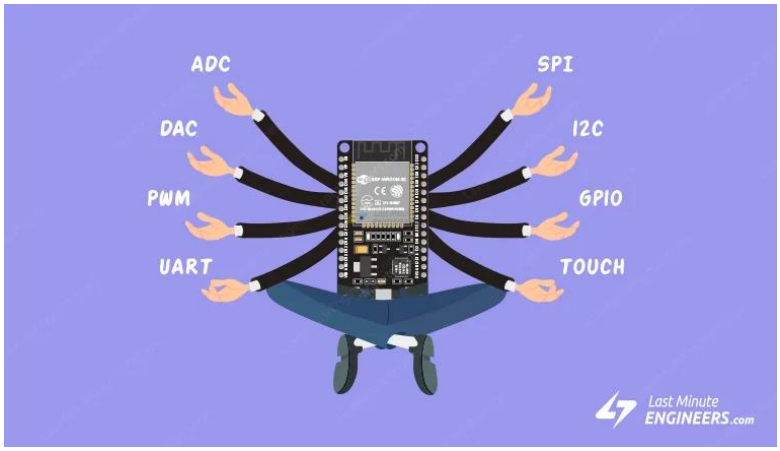
ESP32 Pinout Reference



One of the advantages of the ESP32 is that it has a lot more GPIOs than the ESP8266. You won't have to juggle or multiplex your IO pins. However, there are a few things to keep in mind, so please read the pinout carefully.

Note:

Please note that the following pinout reference is for the popular 30-pin ESP32 devkit v1 development board.

Not every ESP32 development board exposes every pin, but each pin works exactly the same no matter which development board you use.

ESP32 Peripherals and I/O

Although the ESP32 has 48 GPIO pins in total, only 25 of them are broken out to the pin headers on both sides of the development board. These pins can be assigned a variety of peripheral duties, including:

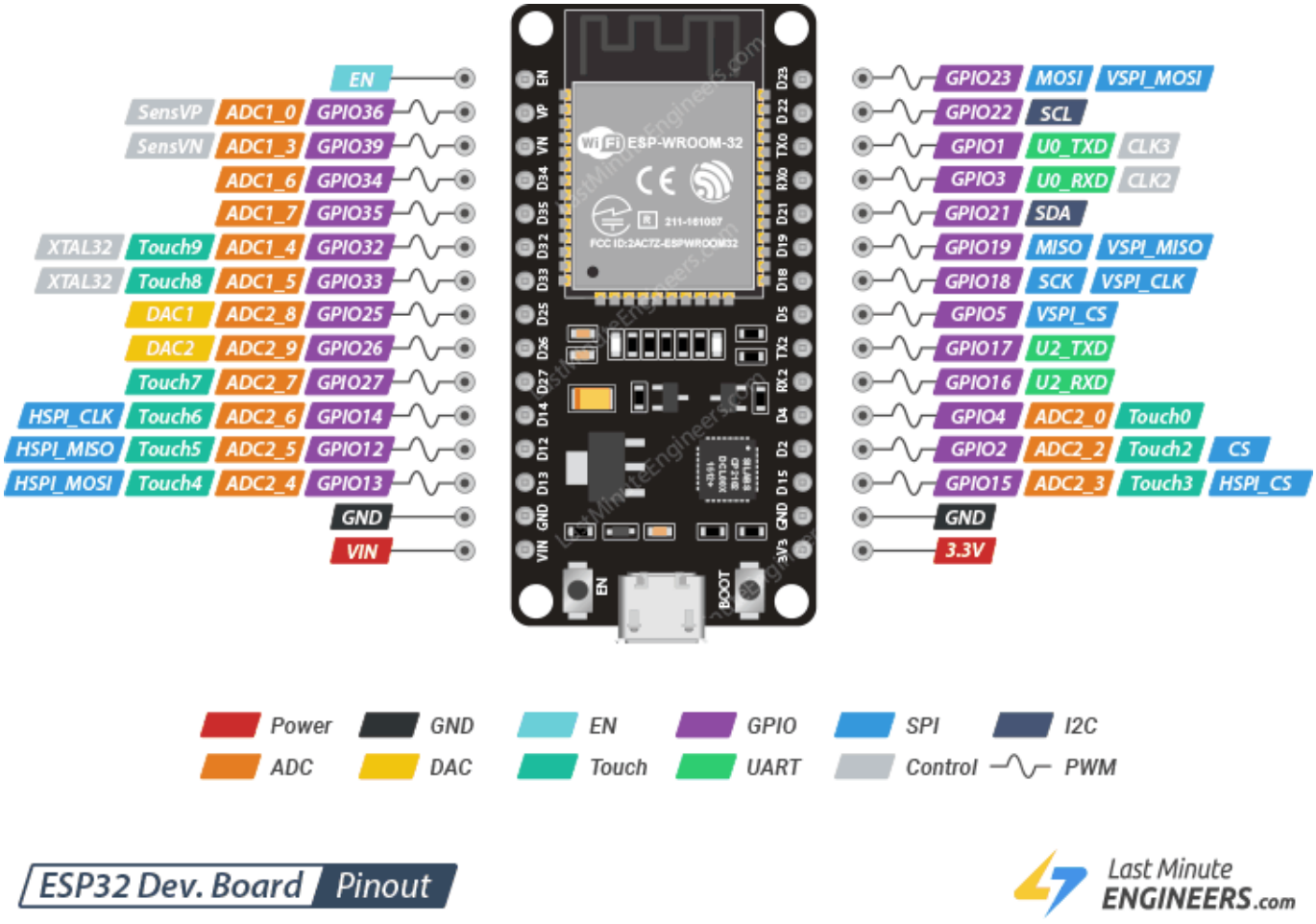
15 ADC channels	15 channels of 12-bit SAR ADC with selectable ranges of 0-1V, 0-1.4V, 0-2V, or 0-4V
2 UART interfaces	2 UART interfaces with flow control and IrDA support
25 PWM outputs	25 PWM pins to control things like motor speed or LED brightness
2 DAC channels	Two 8-bit DACs to generate true analog voltages
SPI, I2C and I2S interface	Three SPI and one I2C interfaces for connecting various sensors and peripherals, as well as two I2S interfaces for adding sound to your project
9 Touch Pads	9 GPIOs with capacitive touch sensing

Thanks to the ESP32's pin multiplexing feature, which allows multiple peripherals to share a single GPIO pin. For example, a single GPIO pin can act as an ADC input, DAC output, or touch pad.

For extensive information about the ESP32, please refer to the datasheet

ESP32 Pinout

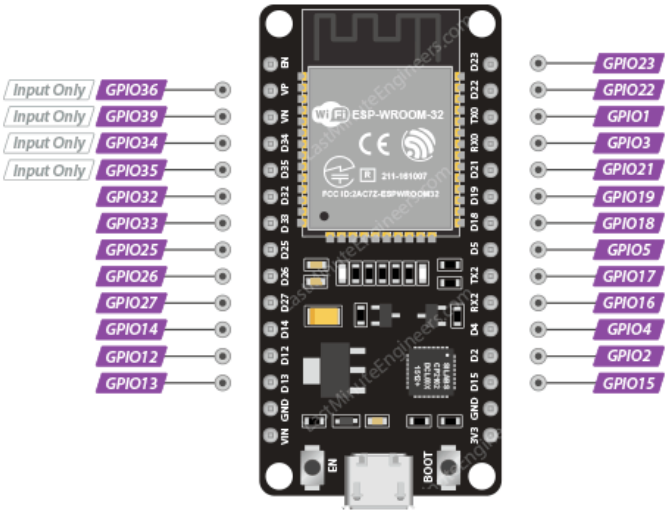
The ESP32 DevKit V1 development board has 30 pins in total. For convenience, pins with similar functionality are grouped together. The pinout is as follows:



Let's take a closer look at the ESP32 pins and their functions one by one.

GPIO Pins

The ESP32 development board has 25 GPIO pins that can be assigned different functions by programming the appropriate registers. There are several kinds of GPIOs: digital-only, analog-enabled, capacitive-touch-enabled, etc. Analog-enabled GPIOs and Capacitive-touch-enabled GPIOs can be configured as digital GPIOs. Most of these digital GPIOs can be configured with internal pull-up or pull-down, or set to high impedance.



Which ESP32 GPIOs are safe to use?

Although the ESP32 has a lot of pins with various functions, some of them may not be suitable for your projects. The table below shows which pins are safe to use and which pins should be used with caution.

✅ – Your top priority pins. They are perfectly safe to use.
⚠️ – Pay close attention because their behavior, particularly during boot, can be unpredictable. Use them only when absolutely necessary.
❌ – It is recommended that you avoid using these pins.

Label	GPIO	Safe to use?	Reason
D0	0	⚠️	must be HIGH during boot and LOW for programming
TX0	1	❌	Tx pin, used for flashing and debugging
D2	2	⚠️	must be LOW during boot and also connected to the on-board LED
RX0	3	❌	Rx pin, used for flashing and debugging
D4	4	✅	
D5	5	⚠️	must be HIGH during boot
D6	6	❌	Connected to Flash memory
D7	7	❌	Connected to Flash memory
D8	8	❌	Connected to Flash memory
D9	9	❌	Connected to Flash memory
D10	10	❌	Connected to Flash memory
D11	11	❌	Connected to Flash memory
D12	12	⚠️	must be LOW during boot
D13	13	✅	
D14	14	✅	
D15	15	⚠️	must be HIGH during boot, prevents startup log if pulled LOW
RX2	16	✅	

TX2	17	✓	
D18	18	✓	
D19	19	✓	
D21	21	✓	
D22	22	✓	
D23	23	✓	
D25	25	✓	
D26	26	✓	
D27	27	✓	
D32	32	✓	
D33	33	✓	
D34	34	!	Input only GPIO, cannot be configured as output
D35	35	!	Input only GPIO, cannot be configured as output
VP	36	!	Input only GPIO, cannot be configured as output
VN	39	!	Input only GPIO, cannot be configured as output

The image below shows which GPIO pins can be used safely.

Input Only GPIOs

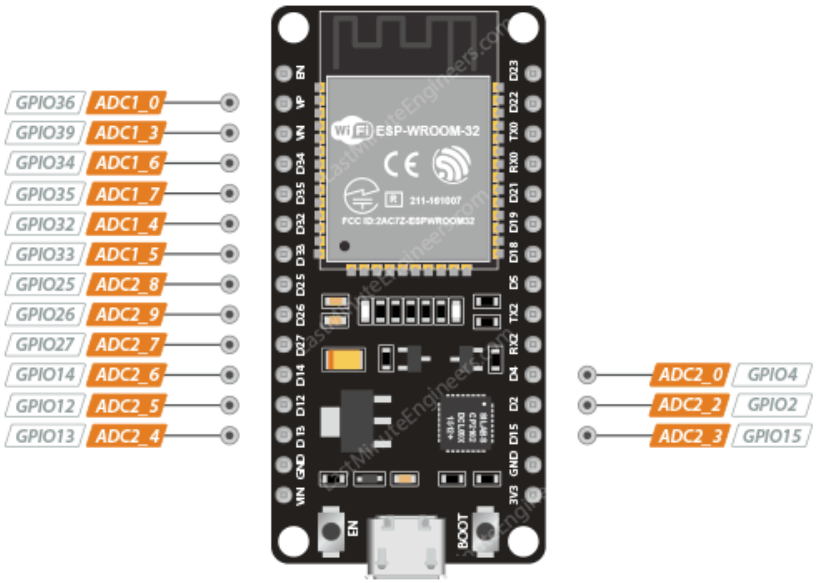
Pins GPIO34, GPIO35, GPIO36(VP) and GPIO39(VN) cannot be configured as outputs. They can be used as digital or analog inputs, or for other purposes. They also lack internal pull-up and pull-down resistors, unlike the other GPIO pins.

ESP32 Interrupt Pins

All GPIOs can be configured as interrupts. For more information, please refer to this tutorial.

ADC Pins

ESP32 integrates two 12-bit SAR ADCs and supports measurements on 15 channels (analog-enabled pins).



The ESP32's ADC is a 12-bit ADC, which means it can detect 4096 (2^12) discrete analog levels. In other words, it will convert input voltages ranging from 0 to 3.3V (operating voltage) into integer values ranging from 0 to 4095. This results in a resolution of 3.3 volts / 4096 units, or 0.0008 volts (0.8 mV) per unit.

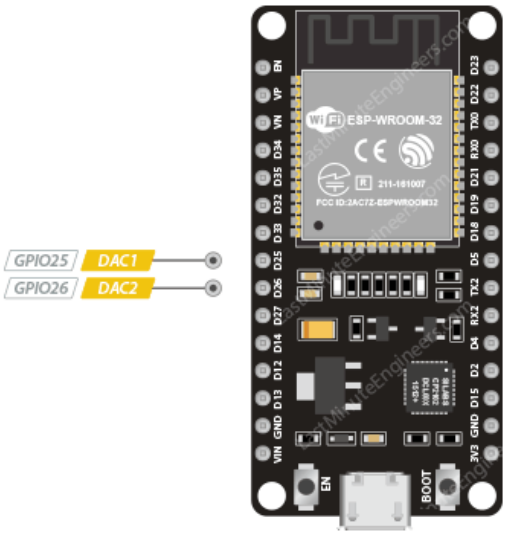
Moreover, the ADC resolution and channel range can be set programmatically.

Warning:

The ADC2 pins cannot be used when Wi-Fi is enabled. If your project requires Wi-Fi, consider using the ADC1 pins instead.

DAC Pins

The ESP32 includes two 8-bit DAC channels for converting digital signals to true analog voltages. It can be used as a "digital potentiometer" to control analog devices.

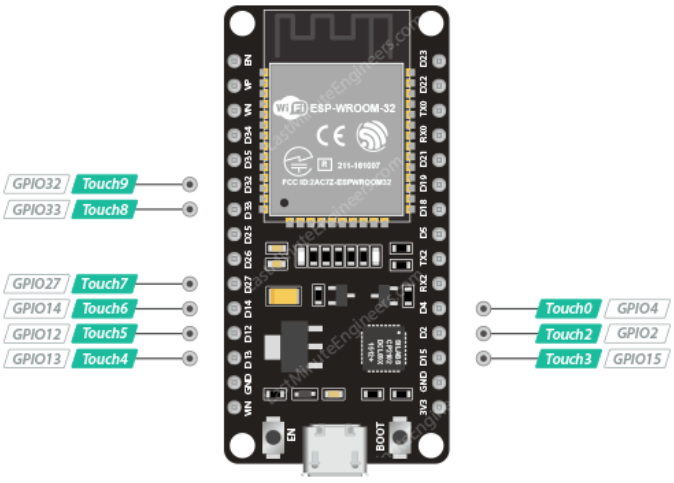


These DACs have an 8-bit resolution, which means that values ranging from 0 to 256 will be converted to an analog voltage ranging from 0 to 3.3V.

The DAC's 8-bit resolution may be insufficient for use in audio applications, in which case an external DAC with a higher resolution (12-24 bits) is preferable.

Touch Pins

The ESP32 has 9 capacitive touch-sensing GPIOs. When a capacitive load (such as a human finger) is in close proximity to the GPIO, the ESP32 detects the change in capacitance.

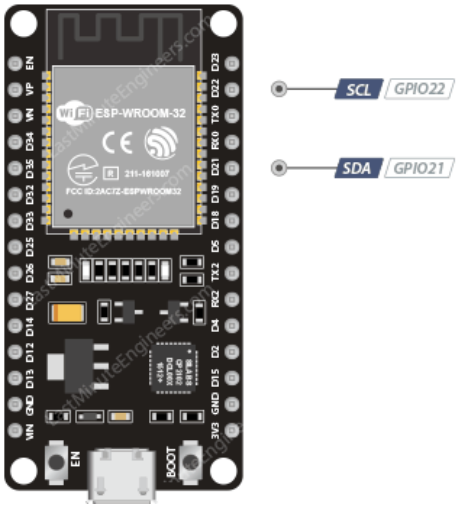


You can make a touch pad by attaching any conductive object to these pins, such as aluminum foil, conductive cloth, conductive paint, and so on. Because of the low-noise design and high sensitivity of the circuit, relatively small pads can be made.

Additionally, these capacitive touch pins can be used to [wake the ESP32 from deep sleep](#).

I2C Pins

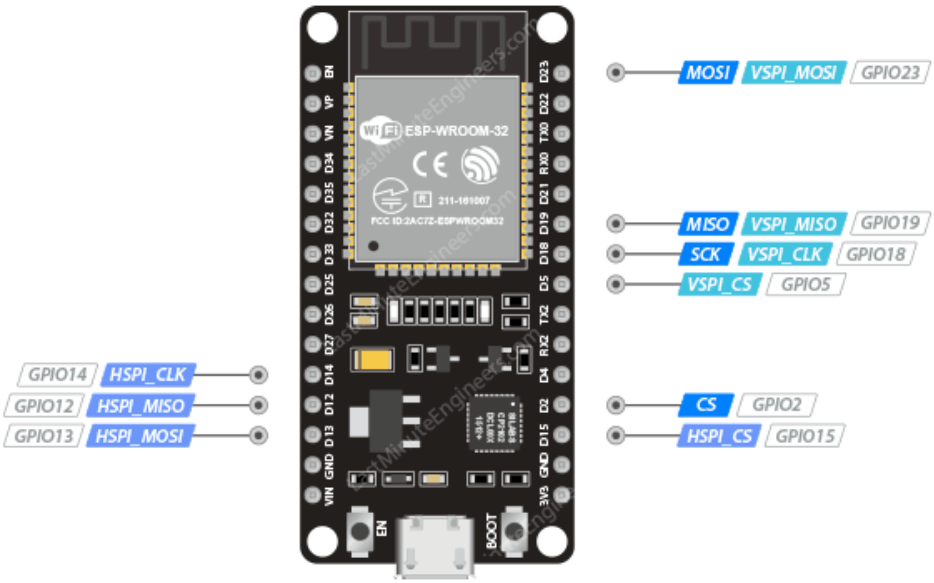
The ESP32 has a single I2C bus that allows you to connect up to 112 sensors and peripherals. The SDA and SCL pins are, by default, assigned to the following pins. However, you can bit-bang the I2C protocol on any GPIO pins with the `wire.begin(SDA, SCL)` command.



SPI Pins

ESP32 features three SPIs (SPI, HSPI, and VSPI) in slave and master modes. These SPIs also support the general-purpose SPI features listed below:

- 4 timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks of 80 MHz
- Up to 64-Byte FIFO



Only VSPI and HSPI are usable SPI interfaces, and the third SPI bus is used by the integrated flash memory chip. VSPI pins are commonly used in standard libraries.

HSPI vs. VSPI

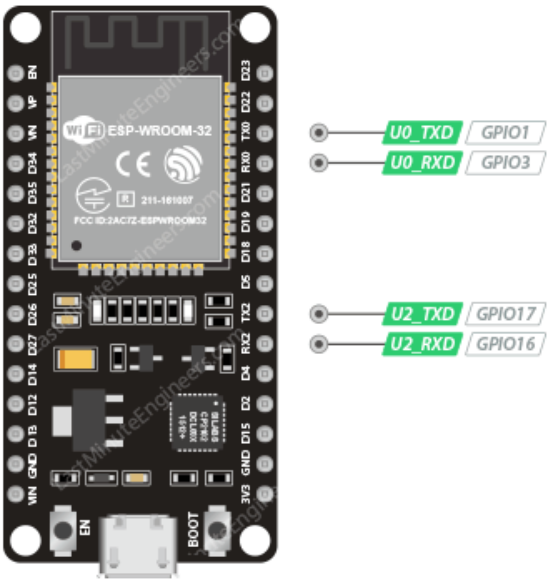
HSPI is sometimes misinterpreted as “Hardware” SPI and VSPI as “Virtual or Software” SPI. In reality, however, they are identical!

As with I2C, you can bit-bang the SPI protocol on any GPIO pins with the `bus.begin(CLK_PIN, MISO_PIN, MOSI_PIN, SS_PIN);` command.

UART Pins

The ESP32 dev. board has three UART interfaces, UART0, UART1, and UART2, that support asynchronous communication (RS232 and RS485) and IrDA at up to 5 Mbps.

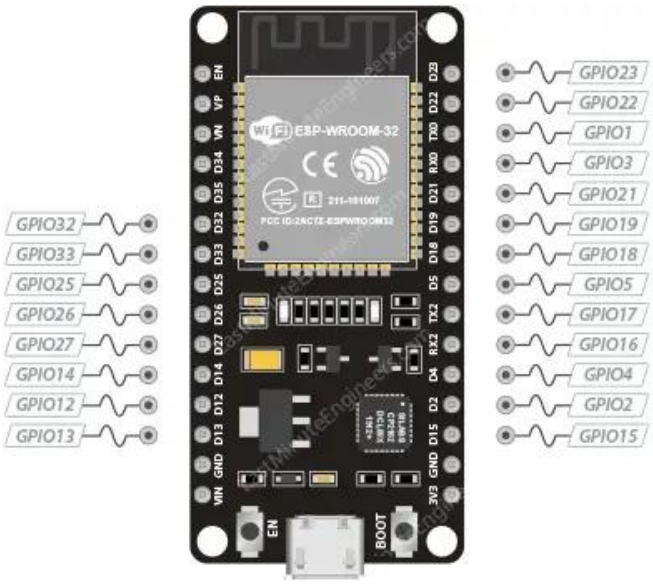
- UART0 pins are connected to the USB-to-Serial converter and are used for flashing and debugging. Therefore, the UART0 pins are not recommended for use.
- UART1 pins are reserved for the integrated flash memory chip.
- UART2, on the other hand, is a safe option for connecting to UART-devices such as GPS, fingerprint sensor, distance sensor, and so on.



In addition, UART provides hardware management of the CTS and RTS signals and software flow control (XON and XOFF) as well.

PWM Pins

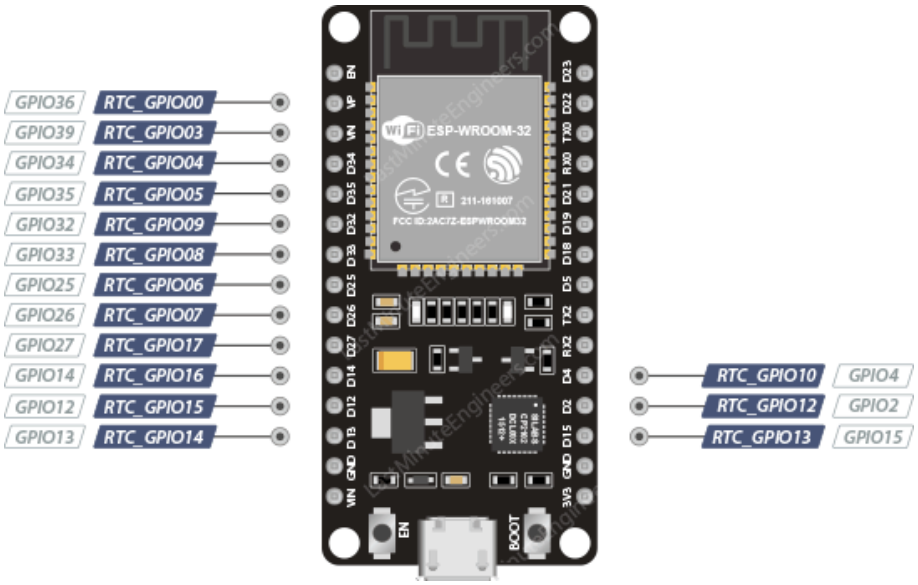
The board has 21 channels (all GPIOs except input-only GPIOs) of PWM pins controlled by a PWM controller. The PWM output can be used for driving digital motors and LEDs.



The PWM controller consists of PWM timers, the PWM operator and a dedicated capture sub-module. Each timer provides timing in synchronous or independent form, and each PWM operator generates a waveform for one PWM channel. The dedicated capture sub-module can accurately capture events with external timing.

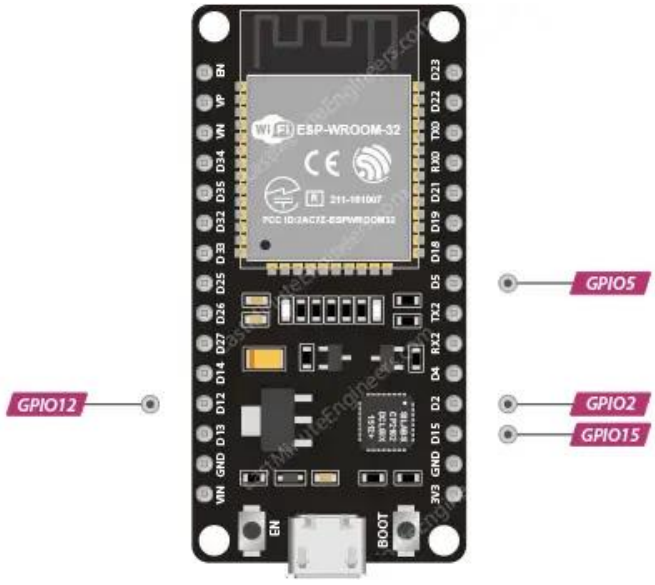
RTC GPIO Pins

Some GPIOs are routed to the RTC low-power subsystem and are referred to as RTC GPIOs. These pins are used to wake the ESP32 from deep sleep when the Ultra-Low Power (ULP) co-processor is running. The GPIOs highlighted below can be used as external wake up sources.



Strapping Pins

There are five strapping pins: GPIO0, GPIO2, GPIO5, GPIO12, and GPIO15.



These pins are used to put the ESP32 into BOOT mode (to run the program stored in the flash memory) or FLASH mode (to upload the program to the flash memory). Depending on the state of these pins, the ESP32 will enter BOOT mode or FLASH mode at power on.

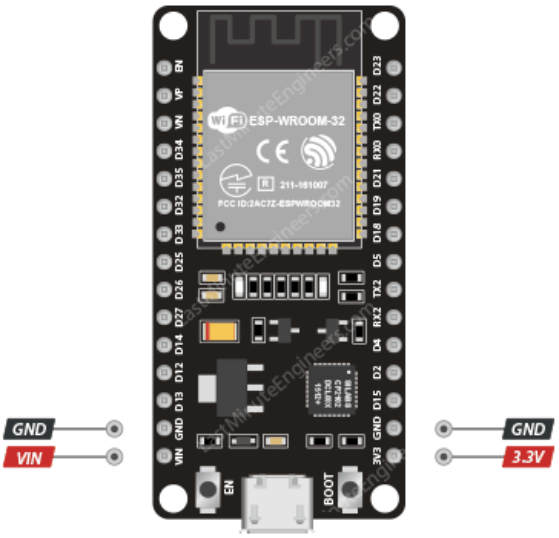
On most development boards with built-in USB/Serial, you don't need to worry about the state of these pins, as the board puts them in the correct state for flashing or boot mode.

However, if peripherals are connected to these pins, you may encounter issues when attempting to upload new code or flash the ESP32 with new firmware, as these peripherals prevent the ESP32 from entering the correct mode.

The strapping pins function normally after reset release, but they should still be used with caution.

Power Pins

There are two power pins: the VIN pin and the 3V3 pin. The VIN pin can be used to directly power the ESP32 and its peripherals if you have a regulated 5V power supply. The 3V3 pin is the output from the on-board voltage regulator; you can get up to 600mA from it. GND is the ground pin.



Enable Pin

The EN pin is the enable pin for the ESP32, pulled high by default. When pulled HIGH, the chip is enabled; when pulled LOW, the chip is disabled.

The EN pin is also connected to a pushbutton switch that can pull the pin LOW and trigger a reset.

Original Article: lastminuteengineers.com/esp32-pinout-reference/