ESP32 - PinOut

The ESP32 chip comes with 48 pins with multiple functions. Not all pins are exposed in all ESP32 development boards, and there are some pins that cannot be used. There are many questions on how to use the ESP32's GPIO. What pins should you use? What pins should you avoid using in your projects? The blog is unique and it has detailed diagrams and guides for the ESP32 GPIOs.

ESP-WROOM-32 CHIP PINOUT

The following figure illustrates the ESP-WROOM-32 chip pinout. Use this diagram if you're using an ESP32 bare chip in your projects. On the bottom of the blog, you can download the PDF with the GPIO - PinOut.

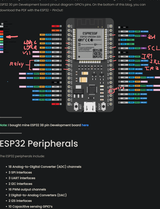
Note: I bought this ESP-WROOM-32 chip here

Note: All ESP32's GPIO are accessible in all development boards, but each specific GPIO works in the same way regardless of the development board you using. If you're just getting started with the GPIO, we recommend reading our Getting Started Guide with the ESP32 Development Board.

At the moment, there are a wide variety of development boards with the ESP32 chip. We differ in the number of accessible GPIOs, the form factor, etc.

ESP-WROOM-32 30 PIN Develepment Board PINOUT

If you're using an ESP32 30 pin Development board, you can use the following GPIO diagram as a reference. ESP32 30 pin Development board has 36 available GPIO pins. On the bottom of the blog, you can download the PDF with the GPIO - PinOut.



Note: I bought this ESP32 30 pin Development board here

ESP-WROOM-32 38 PIN Develepment Board PINOUT

If you're using an ESP32 38 pin Development board, you can use the following GPIO diagram as a reference. ESP32 38 pin Development board pinout diagram GPIO pins. On the bottom of the blog, you can download the PDF with the GPIO - PinOut.

Note: I bought this ESP32 38 pin Development board here

ESP32 Peripherals

The ESP32 peripherals include:

- 18 Analog to Digital Converter (SAC) channels
- 3 SPI Interfaces
- 3 UART Interfaces
- 2 I2C Interfaces
- 16 PWM output channels
- 2 Digital to Analog Converters (DAC)
- 2 I2S Interfaces
- 10 Capacitive sensing GPIO's

The ESP32 Analogue to Digital Converted (ADC) Digital to Analogue Converters features are assigned to specific analogue pins. However, you can assign other analogue pins to UART, I2C, SPI, PWM, etc – but you need to assign them in the code. This is possible and is the ESP32's multiplexing feature.

BEST PINS TO USE – ESP32

Generally speaking, the ESP32 GPIO pins are easy to use. However, not all the pins are perfect. The following table shows what pins are best to use as inputs, outputs and which ones you need to be cautious.

The pins highlighted in **green** can be used. The ones highlighted in **yellow** pins can be used, **but you need to pay attention because they may have unexpected behaviour, mainly at boot.** The pins highlighted in **red** are not recommended to use as inputs or outputs.

GPIO	Input	Output	Notes
0	pulled up	OK	Outputs PWM signal at boot
1	Tx pin	OK	debug output at boot
2	OK	OK	connected to on-board LED
3	OK	Rx pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot
6	X	X	connected to the integrated SPI flash
7	X	X	connected to the integrated SPI flash
8	X	X	connected to the integrated SPI flash connected to the integrated SPI flash
9	X	X	connected to the integrated SPI flash
10	X	X	connected to the integrated SPI flash
11	X	X	connected to the integrated SPI flash
12	OK	OK	
13	OK	OK	outputs PWM signal at boot
14	OK	OK	outputs PWM signal at boot
15	OK	OK	
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK	input only	
35	OK	input only	
36	OK	input only	
39	OK	input only	

Continue reading for a more detail and in-depth analysis of the ESP32 GPIO's and its functions.

Input only pins

GPIO's 34 to 39 are GPIO's – input only pins. These pins don't have internal pull-up or pull-down resistors. They can't be used as outputs, so use these pins only as inputs:

- GPIO 34
- GPIO 35
- GPIO 36
- GPIO 39

SPI flash integrated on the ESP-WROOM-32

GPIO 6 - GPIO 11 are exposed in ESP32 development boards. However, these pins are connected to its integrated SPI flash. Pins GPIO 6 - GPIO 11 are not recommended for other uses. So, don't use these pins if you're using ESP32:

- GPIO 6 (SCK/CLK)
- GPIO 7 (SDO/SD0)
- GPIO 8 (SDI/SD1)
- GPIO 9 (SHD/SD2)
- GPIO 10 (SWP/SD3)
- GPIO 11 (CSC/CMD)

Capacitive touch GPIO's

The ESP32 has **10 internal capacitive touch sensors**. These can sense variations in anything that holds an electrical charge, like a human being touch. So, they can detect variations caused when touching the GPIO pins. Touch sensors can be used to sense liquids, integrated into capacitive pads, and replace mechanical buttons. The capacitive touch pins can also be used to wake up the ESP32 from deep sleep.

These internal touch sensors are connected to these GPIO's:

- T0 (GPIO 4)
- T1 (GPIO 0)
- T2 (GPIO 2)
- T3 (GPIO 15)
- T4 (GPIO 13)
- T5 (GPIO 12)
- T6 (GPIO 14)
- T7 (GPIO 27)
- T8 (GPIO 33)
- T9 (GPIO 32)

Analogue to Digital Converter (ADC)

The ESP32 has **18 ADC input channels**. Within the ESP32DR, only Tx & RX's ADC2. These are the GPIO's that can be used as ADC and respective channels:

- ADC1_CH0 (GPIO 36)
- ADC1_CH1 (GPIO 37)
- ADC1_CH2 (GPIO 38)
- ADC1_CH3 (GPIO 39)
- ADC1_CH4 (GPIO 32)
- ADC1_CH5 (GPIO 33)
- ADC1_CH6 (GPIO 34)
- ADC1_CH7 (GPIO 35)
- ADC2_CH0 (GPIO 4)
- ADC2_CH1 (GPIO 0)
- ADC2_CH2 (GPIO 2)
- ADC2_CH3 (GPIO 15)
- ADC2_CH4 (GPIO 13)
- ADC2_CH5 (GPIO 12)
- ADC2_CH6 (GPIO 14)
- ADC2_CH7 (GPIO 27)
- ADC2_CH8 (GPIO 25)
- ADC2_CH9 (GPIO 26)

How to use the ESP32 ADC pins

Note: ADC2 pins cannot be used when Wi-Fi is used. So, if you're using Wi-Fi, and you're having trouble getting the value from an **ADC2 GPIO**, you may consider using an **ADC1 GPIO** instead. That should solve your problem.

The **ADC input channels** have 12-bit resolution. This means that you can get analogue readings ranging from **0 to 4095**, in which 0 corresponds to 0V and 4095 to 3.3V. You also have the ability to set the resolution of your channel, or the code, as well as the ADC range.

The ESP32 ADC pins don't have a linear behaviour. You probably won't be able to distinguish between 0 and 0.1V, or between 3.3 and 3.3V. You need to keep that in mind when using the ADC pins.

Digital to Analog Converter (DAC)

There are **2 8-bit DAC channels** in the ESP32 to convert digital signals into analogue voltage signal outputs. These are the DAC channels:

- DAC1 (GPIO25)
- DAC2 (GPIO26)

RTC GPIOs

There is **RTC GPIO** support on the ESPS. The GPIO's routed to the **RTC low power subsystem** can be used when the ESP32 is in **deep sleep**. These RTC GPIO's can be used to wake up the ESP32 from deep sleep when the **Ultra Low Power (ULP) co-processor** is running. The following GPIO's can be used as an external wake-up source.

- RTC_GPIO0 (GPIO36)
- RTC_GPIO1 (GPIO37)
- RTC_GPIO2 (GPIO38)
- RTC_GPIO3 (GPIO39)
- RTC_GPIO4 (GPIO34)
- RTC_GPIO5 (GPIO35)
- RTC_GPIO6 (GPIO25)
- RTC_GPIO7 (GPIO26)
- RTC_GPIO8 (GPIO33)
- RTC_GPIO9 (GPIO32)
- RTC_GPIO10 (GPIO4)
- RTC_GPIO11 (GPIO0)
- RTC_GPIO12 (GPIO2)
- RTC_GPIO13 (GPIO15)
- RTC_GPIO14 (GPIO13)
- RTC_GPIO15 (GPIO12)
- RTC_GPIO16 (GPIO14)
- RTC_GPIO17 (GPIO27)

PWM

There are **16 PWM** controller has **16 independent channels** that can be configured to generate PWM signals with different properties. All pins that can act as outputs can be used as PWM pins. **GPIO's 34 to 39 can't generate PWM.**

To set a PWM signal, you need to define these parameters in the code:

- Signal's frequency
- Duty cycle
- PWM channel
- GPIO where you want to output the signal.

I2C

The ESP32 has **two I2C channels**. and any pin can be set as **SDA** or **SCL**. The default I2C pins are:

- GPIO 21 (SDA)
- GPIO 22 (SCL)

SPI

By default, the pin mapping for SPI is:

SPI	MOSI	MISO	CLK	CS
VSPI	GPIO 23	GPIO 19	GPIO 18	GPIO 5
HSPI	GPIO 13	GPIO 12	GPIO 14	GPIO 15

Interrupts

All GPIO's can be configured as interrupts.

Strapping Pins

The ESP32 chip has the following strapping pins:

- GPIO 0
- GPIO 2
- GPIO 5
- GPIO 12 (can be read using boot)
- GPIO 15 (can be read using boot)
- GPIO 15 (can be read using boot)

These are used to control **the ESP32's boot/flash or flashing mode**. On most development boards with built-in pull-up resistors, this will not cause any trouble. However, if you're using strapping pins for other uses, the board pulls the pins in the right way for flashing or boot to work.

However, if you have peripherals connected to these pins, you may have **trouble trying to upload new binaries**. Or, you may have **to remove** it, **reset/flip the board** if you have an I/O or peripherals connected to the strapping pins, and you are getting trouble uploading code or flashing the ESP32. It may be because the peripherals connected to the GPIO's to stress the right mode, after resetting, flashing or booting.

Pins HIGH at Boot

Some GPIO's change to HIGH or output **PWM** signals as soon as boot/reset. This means that if you have outputs connected to these GPIO's, you may get unexpected results when the ESP32 resets or boots.

- GPIO 1
- GPIO 3
- GPIO 5
- GPIO 5 - GPIO 5 (connected to the ESP32 integrated SPI flash memory – not recommended to use)
- GPIO 10
- GPIO 11