

HOME ESP32 ESP8266 ESP32-CAM RASPBERRY PI
MICROPYTHON RPi PICO ARDUINO REVIEWS

Learn ESP32

ESP32
Introduction

ESP32
Arduino IDE

ESP32
Arduino IDE
2.0

VS Code and
PlatformIO

ESP32
Pinout

ESP32 Inputs
Outputs

ESP32 PWM

ESP32
Analog Inputs

ESP32
Interrupts
Timers

ESP32 Pinout Reference: Which GPIO pins should you use?

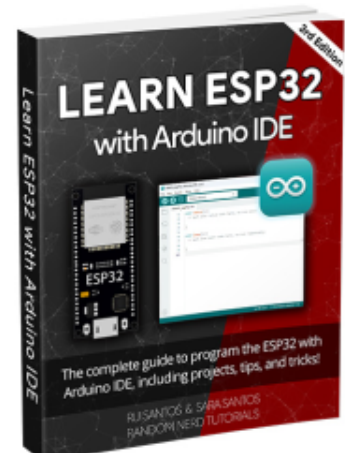
The ESP32 chip comes with 48 pins with multiple functions. Not all pins are exposed in all ESP32 development boards, and some pins cannot be used.

There are many questions on how to use the ESP32 GPIOs. What pins should you use? What pins should you avoid using in your projects? This post aims to be a simple and easy-to-follow reference guide for the ESP32 GPIOs.

The figure below illustrates the ESP-WROOM-32 pinout. You can use it as a reference if you're using an **ESP32 bare chip** to build a custom board:

Affiliate Disclosure:

Random Nerd Tutorials is a participant in affiliate advertising programs designed to provide a means for us to earn fees by linking to Amazon, eBay, AliExpress, and other sites. We might be compensated for referring traffic and business to these companies.



ESP32 Deep
Sleep

Protocols

ESP32 Web
Server

ESP32 LoRa

ESP32 BLE

ESP32 BLE
Client-Server

ESP32
Bluetooth

ESP32
MQTT

ESP32 ESP-
NOW

ESP32 Wi-Fi

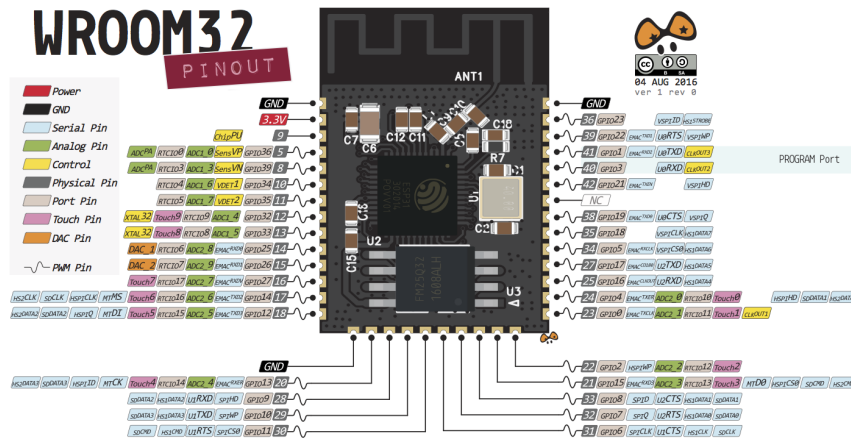
ESP32
WebSocket

ESP32 ESP-
MESH

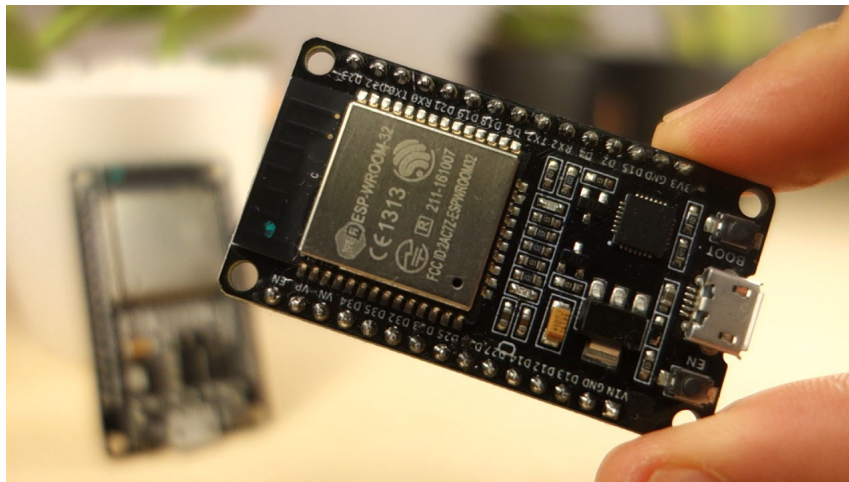
ESP32 Email

ESP32 Text
Messages

ESP32 HTTP
GET POST



Note: not all GPIOs are accessible in all development boards, but each specific GPIO works in the same way regardless of the development board you're using. If you're just getting started with the ESP32, we recommend reading our guide: [Getting Started with the ESP32 Development Board](#).



ESP32 Peripherals

The ESP32 peripherals include:

[Learn ESP32 with Arduino IDE eBook »](#)

Complete guide to program the ESP32 with Arduino IDE!



[SMART HOME with Raspberry Pi, ESP32, and ESP8266 »](#) learn how to build a complete home automation system.

HTTP GET

Web APIs

HTTP POST

Web APIs

Server-Sent

Events

Web Servers

Output Web

Server

PWM Slider

Web Server

PWM

Multiple

Sliders Web

Server

Async Web

Server

Relay Web

Server

Servo Web

Server

DHT Web

Server

BME280

Web Server

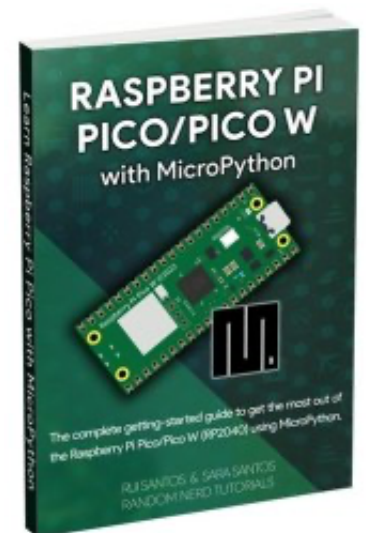
BME680

Web Server

- [18 Analog-to-Digital Converter \(ADC\) 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 GPIOs](#)

The ADC (analog to digital converter) and DAC (digital to analog converter) features are assigned to specific static pins. However, you can decide which pins are UART, I2C, SPI, PWM, etc – you just need to assign them in the code. This is possible due to the ESP32 chip's multiplexing feature.

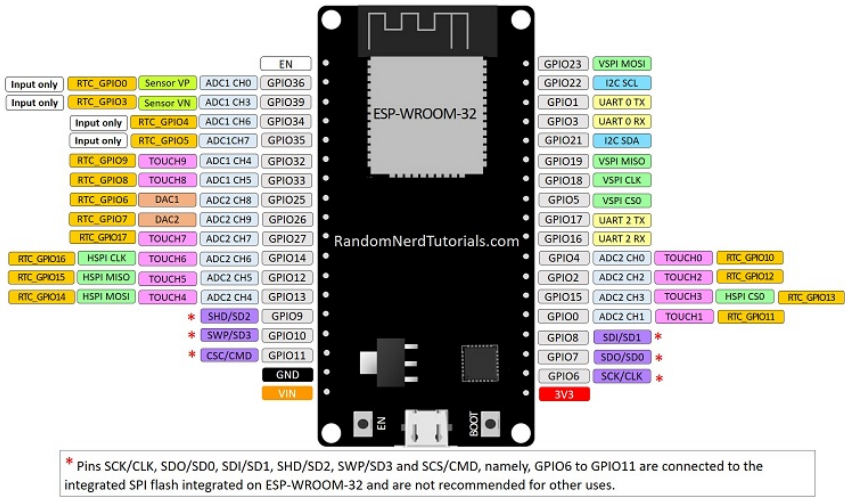
Although you can define the pins properties on the software, there are pins assigned by default as shown in the following figure (this is an example for the [ESP32 DEVKIT V1 DOIT board](#) with 36 pins – the pin location can change depending on the manufacturer).



[Learn Raspberry Pi Pico/Pico W with MicroPython »](#) The complete getting started guide to get the most out of the the Raspberry Pi Pico/Pico W (RP2040) microcontroller board using MicroPython programming language.

- DS18B20
- Web Server
- LoRa Web Server
- Plot/Chart
- Web Server
- Chart
- Multiple Series Web Server
- SPIFFS Web Server
- Thermostat Web Server
- Momentary Switch Web Server
- Physical Button Web Server
- Input Fields Web Server
- Images Web Server
- RGB LED Web Server
- Timer/Pulse Web Server

ESP32 DEVKIT V1 – DOIT
version with 36 GPIOs



Additionally, there are pins with specific features that make them suitable or not for a particular project. 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 are OK to use. The ones highlighted in yellow are OK to use, but you need to pay attention because they may have an unexpected behavior 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, must be LOW to enter flashing mode



Learn LVGL: Build GUIs for ESP32 Projects » Learn how to build Graphical User Interfaces (GUIs) for ESP32 Projects using LVGL (Light Versatile Graphics Library) with the Arduino IDE.

HTTP Auth Web Server	1	TX pin	OK	debug output at boot
MPU-6050 Web Server				connected to on-board LED, must be left floating or LOW to enter flashing mode
MicroSD Card Web Server	2	OK	OK	
Stepper Motor Web Server	3	OK	RX pin	HIGH at boot
Stepper Motor WebSocket	4	OK	OK	
Gauges Web Server	5	OK	OK	outputs PWM signal at boot, strapping pin
DIY Cloud	6	x	x	connected to the integrated SPI flash
ESP32 Weather Station	7	x	x	connected to the integrated SPI flash
Control GPIOs	8	x	x	connected to the integrated SPI flash
View Sensor Readings	9	x	x	connected to the integrated SPI flash
ESP32 MySQL	10	x	x	connected to the integrated SPI flash
ESP32 PHP Email				connected to the

ESP32 SIM800L	11	x	x	integrated SPI flash
Cloud Node- RED Dashboard	12	OK	OK	boot fails if pulled high, strapping pin
Cloud MQTT Broker	13	OK	OK	
ESP32 Cloud MQTT	14	OK	OK	outputs PWM signal at boot
ESP-NOW	15	OK	OK	outputs PWM signal at boot, strapping pin
ESP-NOW Introduction	16	OK	OK	
ESP-NOW Two-Way	17	OK	OK	
ESP-NOW One-to-Many	18	OK	OK	
ESP-NOW	19	OK	OK	
Many-to-One	21	OK	OK	
ESP-NOW + Wi-Fi Web Server	22	OK	OK	
	23	OK	OK	
Firebase	25	OK	OK	
Firestore Realtime Database	26	OK	OK	
Firestore	27	OK	OK	
Web App	32	OK	OK	

Firestore Authentication

Firestore BME280

Firestore Web App Sensor Readings

33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only
39	OK		input only

Firestore ESP32 Data Logging

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

Modules

ESP32 Relay Module

ESP32 DC Motors

ESP32 Servo

Input only pins

GPIOs 34 to 39 are GPIOs – 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

ESP32 Stepper Motor

ESP32 MicroSD Card

SPI flash integrated on the ESP-WROOM-32

ESP32 MicroSD Card Data Logging

GPIO 6 to GPIO 11 are exposed in some ESP32 development boards. However, these pins are connected to the integrated SPI flash on the ESP-

ESP32 PIR
ESP32 HC-SR04

WROOM-32 chip and are not recommended for other uses. So, don't use these pins in your projects:

ESP32 I2C Multiplexer

Sensors

ESP32 DHT11/DHT22

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

ESP32 BME280

ESP32 BME680

ESP32 DS18B20

ESP32 Multiple DS18B20

ESP32 BMP180

ESP32 BMP388

MQTT DHT11/DHT22

MQTT BME280

- T0 (GPIO 4)
- T1 (GPIO 0)
- T2 (GPIO 2)
- T3 (GPIO 15)
- T4 (GPIO 13)

Capacitive touch GPIOs

The ESP32 has 10 internal capacitive touch sensors. These can sense variations in anything that holds an electrical charge, like the human skin. So they can detect variations induced when touching the GPIOs with a finger. These pins can be easily integrated into capacitive pads and replace mechanical buttons. The capacitive touch pins can also be used to [wake up the ESP32 from deep sleep](#).

Those internal touch sensors are connected to these GPIOs:

- MQTT
- BME680
- MQTT
- DS18B20
- ESP32 MPU-6050
- T5 (GPIO 12)
- T6 (GPIO 14)
- T7 (GPIO 27)
- T8 (GPIO 33)
- T9 (GPIO 32)

Displays

Learn how to use the touch pins with Arduino IDE: [ESP32 Touch Pins with Arduino IDE](#)

- ESP32 OLED
- ESP32 LCD
- OLED
- Temperature

Analog to Digital Converter (ADC)

The ESP32 has 18 x 12 bits ADC input channels (while the [ESP8266 only has 1x 10 bits ADC](#)). These are the GPIOs that can be used as ADC and respective channels:

ESP32 Features

- ESP32 Hall Sensor
- ESP32 Touch Sensor
- ESP32 I2C
- ESP32 Flash Memory
- ESP32 Dual Core
- Useful Guides
- ESP32 Troubleshooti
- 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)

- ng
 - ADC2_CH6 (GPIO 14)
- ESP32
 - ADC2_CH7 (GPIO 27)
- Access Point
 - ADC2_CH8 (GPIO 25)
 - ADC2_CH9 (GPIO 26)

ESP32 Fixed
IP Address

Learn how to use the ESP32 ADC pins:

ESP32 MAC
Address

- [ESP32 ADC Pins with Arduino IDE](#)
- [ESP32 ADC Pins with MicroPython](#)

ESP32
Hostname

ESP32 OTA

ESP32 OTA
Arduino

ESP32 OTA
VS Code

ESP32 Solar
Panels

ESP32 Alexa

ESP32 Install
SPIFFS

ESP32 Time
and Date

ESP32
Epoch Time

ESP32
Google
Sheets

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 a 12-bit resolution. This means that you can get analog readings ranging from 0 to 4095, in which 0 corresponds to 0V and 4095 to 3.3V. You can also set the resolution of your channels on the code and the ADC range.

The ESP32 ADC pins don't have a linear behavior. You'll probably won't be able to distinguish between 0 and 0.1V, or between 3.2 and 3.3V. You need to keep that in mind when using the ADC pins. You'll get a behavior similar to the one shown in the following figure.

ESP32 Email
Alert

ESP32
ThingSpeak

Weather
Station
Shield

ESP32 IoT
Shield

ESP32
Weather
Station PCB

ESP32 Wi-Fi
Manager

VS Code and
PlatformIO

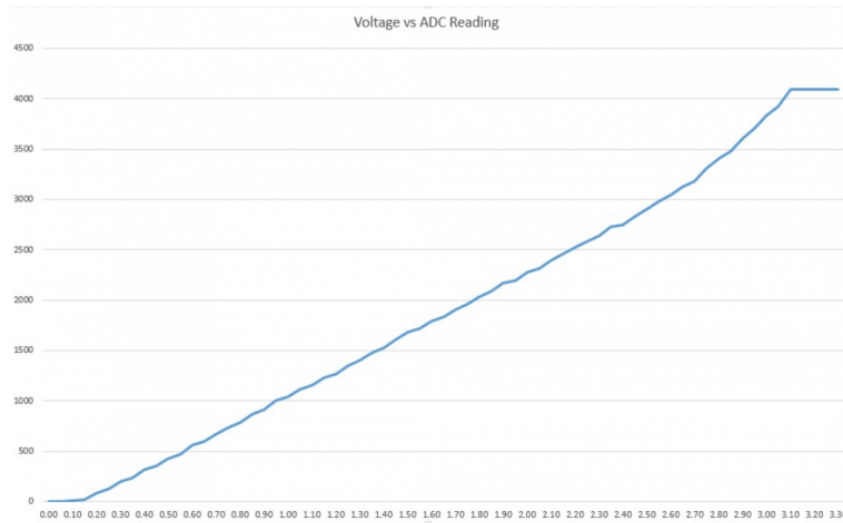
VS Code
SPIFFS

VS Code
Workspaces

Save Data
Preferences
Library

Reconnect to
Wi-Fi

Useful Wi-Fi
Functions



[View source](#)

Digital to Analog Converter (DAC)

There are 2 x 8 bits DAC channels on the ESP32 to convert digital signals into analog voltage signal outputs. These are the DAC channels:

- DAC1 (GPIO25)
- DAC2 (GPIO26)

RTC GPIOs

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

Other Projects

[up source.](#)

Telegram Control Outputs	<ul style="list-style-type: none">▪ RTC_GPIO0 (GPIO36)▪ RTC_GPIO3 (GPIO39)▪ RTC_GPIO4 (GPIO34)▪ RTC_GPIO5 (GPIO35)
Telegram Sensor Readings	<ul style="list-style-type: none">▪ RTC_GPIO6 (GPIO25)▪ RTC_GPIO7 (GPIO26)▪ RTC_GPIO8 (GPIO33)
Telegram Detect Motion	<ul style="list-style-type: none">▪ RTC_GPIO9 (GPIO32)▪ RTC_GPIO10 (GPIO4)▪ RTC_GPIO11 (GPIO0)
Telegram Group	<ul style="list-style-type: none">▪ RTC_GPIO12 (GPIO2)▪ RTC_GPIO13 (GPIO15)▪ RTC_GPIO14 (GPIO13)
ESP32 Status PCB	<ul style="list-style-type: none">▪ RTC_GPIO15 (GPIO12)▪ RTC_GPIO16 (GPIO14)
ESP32 BMP388	<ul style="list-style-type: none">▪ RTC_GPIO17 (GPIO27)
Datalogger	Learn how to use the RTC GPIOs to wake up the ESP32 from deep sleep: ESP32 Deep Sleep with Arduino IDE and Wake Up Sources
ESP32 Web Serial	

ESP32 Door Monitor

PWM

ESP32 Door Telegram

The ESP32 LED 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 (GPIOs 34 to 39 can't generate PWM).

ESP32 NTP Timezones

ESP32 Boards

ESP32
Camera

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

ESP32 LoRa

ESP32 OLED

ESP32

SIM800L

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

Learn More

Learn how to use ESP32 PWM with Arduino

Learn ESP32

IDE: [ESP32 PWM with Arduino IDE](#)

Learn

ESP8266

I2C

Learn

ESP32-CAM

The ESP32 has two I2C channels and any pin can be set as SDA or SCL. When using the ESP32 with the Arduino IDE, the default I2C pins are:

Learn

MicroPython

Learn

Arduino

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

Build Web

Servers

eBook

If you want to use other pins when using the wire library, you just need to call:

Smart Home

eBook

```
Wire.begin(SDA, SCL);
```

Firestore

Web App

eBook

Learn more about I2C communication protocol with the ESP32 using Arduino IDE:
[ESP32 I2C Communication \(Set Pins, Multiple Bus Interfaces and Peripherals\)](#)

ESP32

Premium

Course

More I2C Tutorials with the ESP32:

- [ESP32 I2C Master and Slave \(I2C Communication Between Two ESP32\) – Arduino IDE](#)
- [ESP32: I2C Scanner \(Arduino IDE\) – Finding the Address of I2C Devices](#)
- [Guide for TCA9548A I2C Multiplexer with the ESP32](#)

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

Learn more about SPI communication protocol with the ESP32 using Arduino IDE:
[ESP32 SPI Communication: Set Pins, Multiple SPI Bus Interfaces, and Peripherals \(Arduino IDE\)](#)

UART

The ESP32 supports up to three UART interfaces:

UART0, **UART1**, and **UART2**, depending on the ESP32 board model you're using.

- **UART0** is usually reserved for communication with the serial monitor during upload and debugging. However, you can also use it for communication with other devices after uploading the code if the Serial Monitor is not needed.
- **UART1** and **UART2**: available to communicate with external devices.

Like I2C and SPI, these UART pins can be mapped to any GPIO pin on the ESP32. However, they have a default pin assignment on most board models.

For most ESP32 boards the UART pin assignment is as follows:

UART Port	TX	RX	Remarks
UART0	GPIO 1	GPIO 3	Used for Serial Monitor and uploading code; Can be assigned to other GPIOs;
UART1	GPIO 10	GPIO 9	<u>Must</u> be assigned to other GPIOs

UART2	GPIO 17	GPIO 16	Can be assigned to other GPIOs
--------------	------------	------------	-----------------------------------

About UART1 (GPIO 9 and GPIO10) – these GPIOs are connected to the ESP32 SPI flash memory, so you can't use them like that. To use UART1 to communicate with other devices, you must define different pins using the `HardwareSerial` library.

Learn more about UART with the ESP32:

[ESP32 UART Communication \(Serial\): Set Pins, Interfaces, Send and Receive Data \(Arduino IDE\)](#)

If you're using an [ESP32-S3](#), the assignment is completely different. Check out the [ESP32-S3 pinout here](#).

Interrupts

All GPIOs can be configured as interrupts.

Learn how to use interrupts with the ESP32:

- [ESP32 interrupts with Arduino IDE](#)
- [ESP32 interrupts with MicroPython](#)

Strapping Pins

The ESP32 chip has the following strapping pins:

- GPIO 0 (must be LOW to enter boot mode)
- GPIO 2 (must be floating or LOW during boot)
- GPIO 4
- GPIO 5 (must be HIGH during boot)
- GPIO 12 (must be LOW during boot)
- GPIO 15 (must be HIGH during boot)

These are used to put the ESP32 into bootloader or flashing mode. On most development boards with built-in USB/Serial, you don't need to worry about the state of these pins. The board puts the pins in the right state for flashing or boot mode. More information on the [ESP32 Boot Mode Selection](#) can be found [here](#).

However, if you have peripherals connected to those pins, you may have trouble trying to upload new code, flashing the ESP32 with new firmware, or resetting the board. If you have some peripherals connected to the strapping pins and you are getting trouble uploading code or flashing the ESP32, it may be because those peripherals are preventing the ESP32 from entering the right mode. Read the [Boot Mode Selection documentation](#) to guide you in the right direction. After resetting, flashing, or booting, those pins work as expected.

Pins HIGH at Boot

Some GPIOs change their state to HIGH or output PWM signals at boot or reset. This means that if you have outputs connected to these GPIOs you may get unexpected results when the ESP32 resets or boots.

- GPIO 1
- GPIO 3
- GPIO 5
- GPIO 6 to GPIO 11 (connected to the ESP32 integrated SPI flash memory – not recommended to use).
- GPIO 14
- GPIO 15

Enable (EN)

Enable (EN) is the 3.3V regulator's enable pin. It's pulled up, so connect to ground to disable the 3.3V regulator. This means that you can use this pin connected to a pushbutton to restart your ESP32, for example.

GPIO current drawn

The absolute maximum current drawn per GPIO is 40mA according to the “Recommended Operating Conditions” section in the ESP32 datasheet.

ESP32 Built-In Hall Effect

Sensor

The ESP32 also features a [built-in hall effect sensor](#) that detects changes in the magnetic field in its surroundings.

Wrapping Up

We hope you've found this reference guide for the ESP32 GPIOs useful. If you have more tips about the ESP32 GPIOs, please share by writing a comment down below.

If you're just getting started with the ESP32, we have some great content to get started:

- [Learn ESP32 with Arduino IDE](#)
- [Getting Started with the ESP32 Development Board](#)
- [20+ ESP32 Projects and Tutorials](#)
- [ESP32 Web Server Tutorial](#)
- [ESP32 vs ESP8266 – Pros and Cons](#)

Thanks for reading.