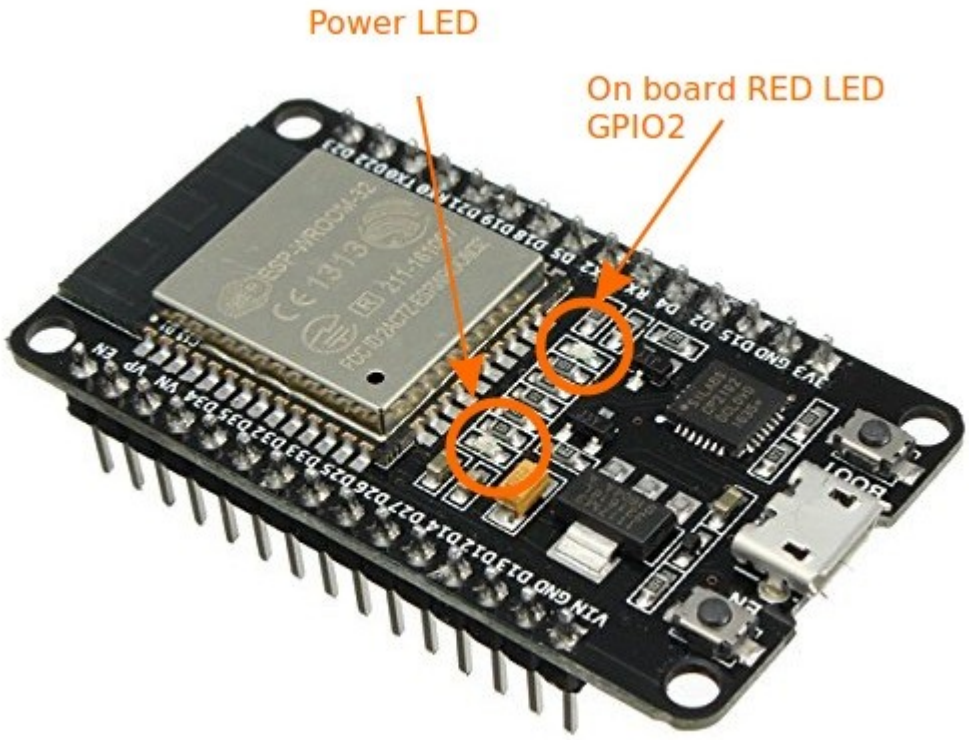
ESP32 DevKit ESP32-WROOM GPIO Pinout

ESP32ESP32, Pinout

Introduction

ESP32-WROOM-32 is a powerful, generic Wi-Fi+BT+BLE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.

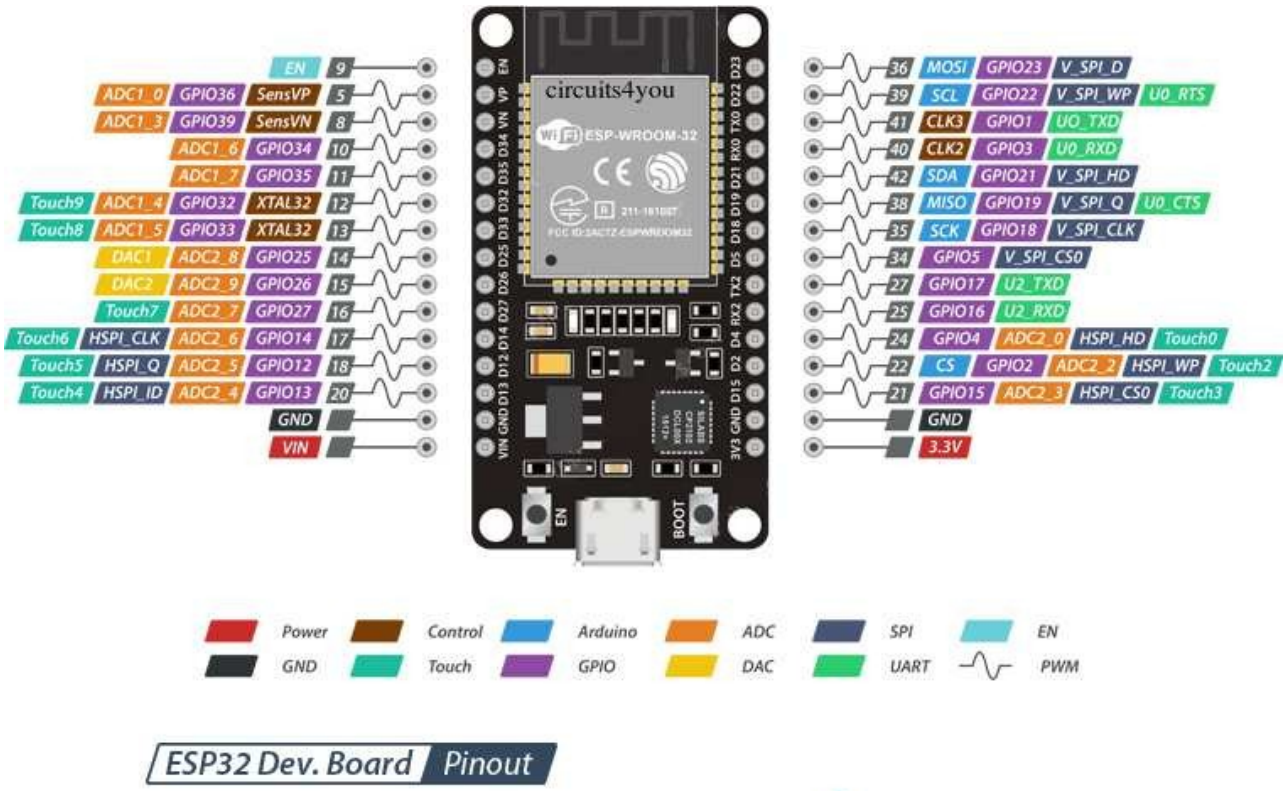


At the core of this module is the ESP32-D0WDQ6 chip*. The chip embedded is designed to be scalable and adaptive. There are two CPU cores that can be individually controlled, and the CPU clock frequency is adjustable from 80 MHz to 240 MHz. The user may also power off the CPU and make use of the low-power co-processor to constantly monitor the peripherals for changes or crossing of thresholds. ESP32 integrates a rich set of peripherals, ranging from capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I2S and I2C.

For more details Read ESP32-WROOM32 Data sheet

The integration of Bluetooth, Bluetooth LE and Wi-Fi ensures that a wide range of applications can be targeted, and that the module is future proof: using Wi-Fi allows a large physical range and direct connection to the internet through a Wi-Fi router, while using Bluetooth allows the user to conveniently connect to the phone or broadcast low energy beacons for its detection. The sleep current of the ESP32 chip is less than 5 µA, making it suitable for battery powered and wearable electronics applications. ESP32 supports a data rate of up to 150 Mbps, and 20.5 dBm output power at the antenna to ensure the widest physical range. As such the chip does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity. The operating system chosen for ESP32 is freeRTOS with LwIP; TLS 1.2 with hardware acceleration is built in as well. Secure (encrypted) over the air (OTA) upgrade is also supported, so that developers can continually upgrade their products even after their release.

ESP32 WROOM32 DevKit Pinout



ESP32 Peripherals Features

- 18 Analog-to-Digital Converter (ADC) channels
- 10 Capacitive sensing GPIOs
- 3 UART interfaces
- 3 SPI interfaces
- 2 I2C interfaces
- 16 PWM output channels
- 2 Digital-to-Analog Converters (DAC)
- 2 I2S interfaces

GPIO Pins

ESP32 Wroom32 DevKit has total 25 GPIOs out of that few pins are Input only Pins,

Input Only Pins

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

Not all pins have input pullup, you need external pullup on these pins when using as input pullup.

Pins with internal pull up INPUT_PULLUP

- GPIO14
- GPIO16
- GPIO17
- GPIO18
- GPIO19
- GPIO21
- GPIO22
- GPIO23

Pins without internal pull up

- GPIO13
- GPIO25
- GPIO26
- GPIO27
- GPIO32
- GPIO33

In arduino to use these pins you can simply use common commands

Example: To make GPIO22 as input and GPIO23 as output

pinMode(22,INPUT_PULLUP);

pinMode(23,OUTPUT);

digitalWrite(23,HIGH);

Analog Input Pins

Note that only a subset of ADC pins and functions are exposed. First, the supplied drivers expose only ADC1. The board layout of the ESP32-DevKitC only exposes some of the pins. Specifically, the following are exposed: **ADC1_CH0 , ADC1_CH3 , ADC1_CH4 , ADC1_CH5 , ADC1_CH6 and ADC1_CH7 .**

see ESP32 Analog Read Example

Analog to digital conversion is the ability to read a voltage level found on a pin between 0 and some maximum value and convert that analog value into a digital representation. Varying the voltage applied to the pin will change the value read. The ESP32 has an analog to digital converter built into it with a **resolution of up to 12 bits which is 4096** distinct values. What that means is that 0 volts will produce a digital value of 0 while the maximum voltage will produce a **digital value of 4095** and voltage ranges between these will produce a correspondingly scaled digital value. One of the properties on the analog to digital converter channels is attenuation. This is a voltage scaling factor. Normally the **input range is 0-1V** but with different attenuations we can scale the input voltage into this range. The available scales beyond the 0-1V include 0-1.34V, 0-2V and 0-3.6V.

Capacitive touch GPIOs

ESP32 Capacitive Touch Example Code

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:

- 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)

Digital to Analog Converter (DAC)

Example code is available here

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 up source.

- RTC_GPIO0 (GPIO36)
- 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

PWM Example Code is here

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 (Input only pin GPIOs 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.

Serial

Hardware Serial2 Example Code

ESP32 has three serial ports

First Serial RX0, TX0 is used for programming,

- **GPIO3 (U0RXD)**
- **GPIO1(U0TXD)**

Another Serial port is available on

- **GPIO16 (U2RXD).**
- **GIIO17 (U2TXD).**

When programming it is named as Serial2.

I2C

When using the ESP32 with the Arduino IDE, you should use the ESP32 I2C default pins (supported by the Wire library):

- 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 GPIOs can be configured as interrupts.

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.

GPIO current drawn

The absolute maximum current drawn per GPIO is source 40mA and sink 28mAmp according to the "Recommended Operating Conditions" section in the ESP32 datasheet.

Ref : https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/