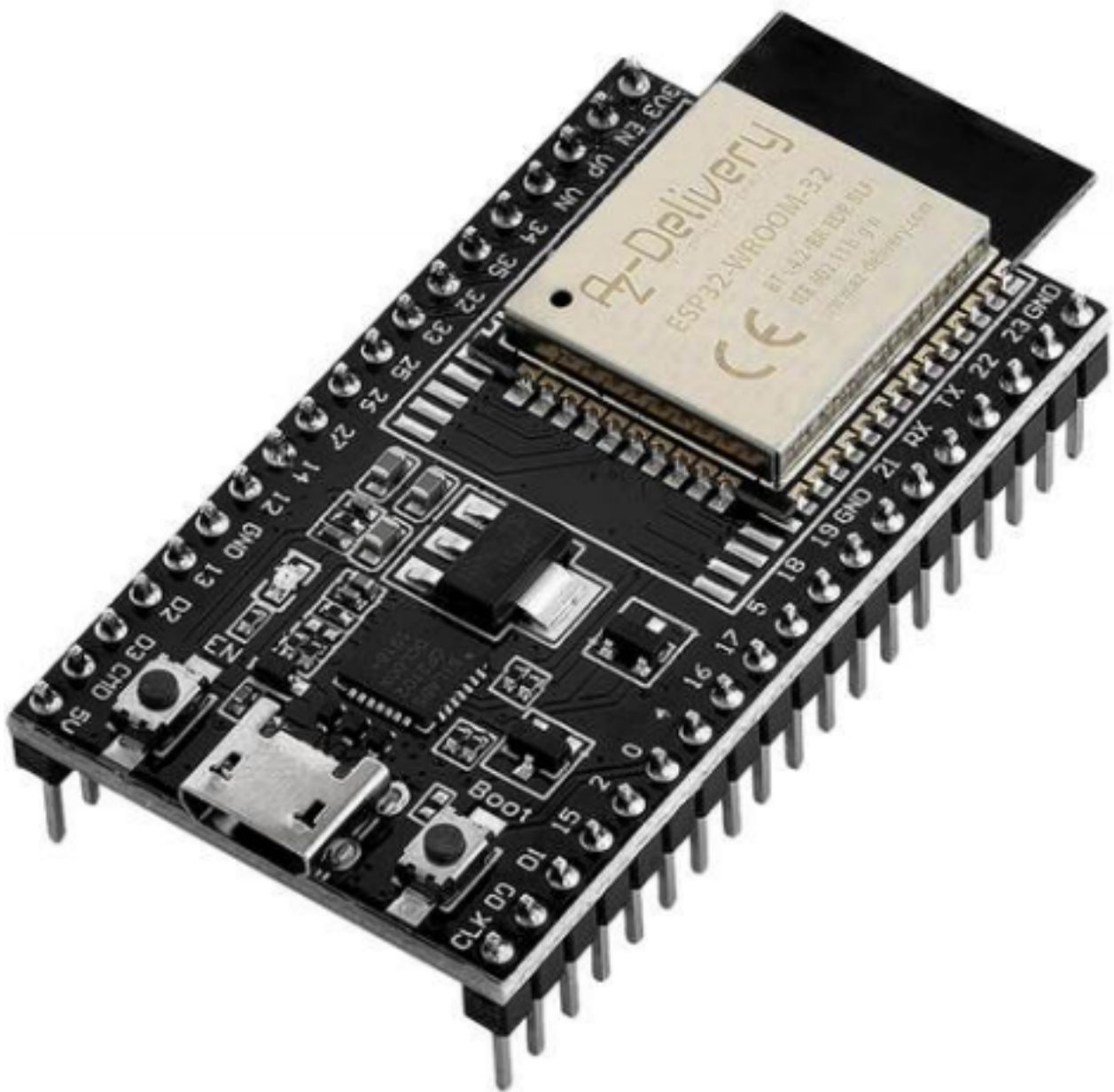




## Welcome!

Thank you for purchasing our *AZ-Delivery ESP-32 Dev Kit C V4*. On the following pages, we will introduce you to how to use and set-up this handy device.

**Have fun!**





## Table of Contents

<b>Introduction</b>	<b>3</b>
<b>Specifications</b>	<b>4</b>
<b>ESP32 Dev Kit C V4</b>	<b>5</b>
<b>Pinout</b>	<b>7</b>
<b>Input / Output pins description</b>	<b>8</b>
<b>Capacitive Touch sensor pins</b>	<b>9</b>
<b>Analog to Digital converter pins</b>	<b>10</b>
<b>Real Time Clock GPIO pins</b>	<b>11</b>
<b>SPI interface pins</b>	<b>13</b>
<b>Strapping Pins</b>	<b>13</b>
<b>Pins HIGH at Boot</b>	<b>14</b>
<b>Enable (EN)</b>	<b>14</b>
<b>USB to Serial communication</b>	<b>15</b>
<b>WiFi Communication</b>	<b>16</b>
<b>Bluetooth Communication</b>	<b>17</b>
<b>Other features</b>	<b>19</b>
<b>How to set-up Arduino IDE</b>	<b>20</b>
<b>Additional setup</b>	<b>24</b>
<b>ESP32 Dev Kit C V4 wiring example</b>	<b>29</b>
<b>Sketch examples</b>	<b>30</b>
<b>PWM - Pulse Width Modulation</b>	<b>31</b>



## Introduction

The ESP32 Dev Kit C V4 is a development board created around ESP32 Wroom 32D chip, containing voltage regulator and USB programmer circuit for ESP32 chip, and a few many other features.

For application development there is a choice between Arduino IDE or ESP-IDF (Native platform). Mostly users choose the Arduino IDE because of its simplicity and compatibility. The user community is very active and supports platforms such as ESP32.

ESP32 Dev Kit C V4 comes with a pre-installed firmware which allows to work with the interpreted language, sending commands through the serial port (*CP2102* chip). The ESP32 Dev Kit C V4 board is one of the most used platforms for Internet of Things (IoT) projects.

The ESP32 Dev Kit C V4 board is specially designed to work on breadboard. It has a voltage regulator that allows it to feed directly from the USB port. The input/output pins work at 3.3V. The *CP2102* chip is responsible for USB to serial communication.



### Specifications

Power supply voltage (USB)	5V DC
Input/Output voltage	3.3V DC
Operating current required	min. 500mA
SoC	ESP32-WROOM-32D
CPU	Xtensa® single-dual-core 32-bit LX6
Clock frequency range	80MHz / 240MHz
RAM	512kB
External flash memory	4MB
I/O pins	34
ADC channels	18
ADC Resolution	12-bit
DAC channels	2
DAC Resolution	8-bit
Communication interfaces	SPI, I2C, I2S, CAN, UART
Wi-Fi protocols	802.11 b/g/n (802.11n up to 150 Mbps)
Wi-Fi frequency	2.4 GHz - 2.5 GHz
Bluetooth	V4.2 - BLE and Classic Bluetooth
Wireless antenna	PCB



## **ESP32 Dev Kit C V4**

The ESP32 WROOM 32D series of Wi-Fi chips is produced by Espressif Systems. ESP32 WROOM-32D is an affordable Wi-Fi module suited for DIY projects in the Internet of Things (IoT) field. This module comes with many GPIOs and support for a variety of protocols like SPI, I2C, I2S, UART, and more. The best part is that it comes with wireless networking included, which makes it different to other microcontrollers. This means that it can easily control and monitor devices remotely via Wi-Fi and Bluetooth® at an affordable price.

ESP32 WROOM-32D is a system-on-chip (SoC) integrating a 32-bit Tensilica microcontroller, standard digital peripheral interfaces, antenna switches, RF balun, power amplifier, low noise receive amplifier, filters and power management modules into a small package. It provides 2.4GHz Wi Fi (802.11 b/g/n, supporting speeds up to 150MB/s), BLE and classic Bluetooth® wireless communication, 34 I/O pins, I2C and I2S interfaces, ADC (analog to digital conversion), DAC (digital to analog conversion), SPI interface, UART on dedicated pins, and PWM (Pulse Width Modulation).

The processor core, called *LX6* by Espressif, is based on Xtensa® dual core 32-bit LX6 processor controller and runs at frequency range between 80-240MHz. It has a 448kB boot ROM, 520kB of on-chip SRAM, and 4MB of external flash memory which can be accessed through SPI interface.

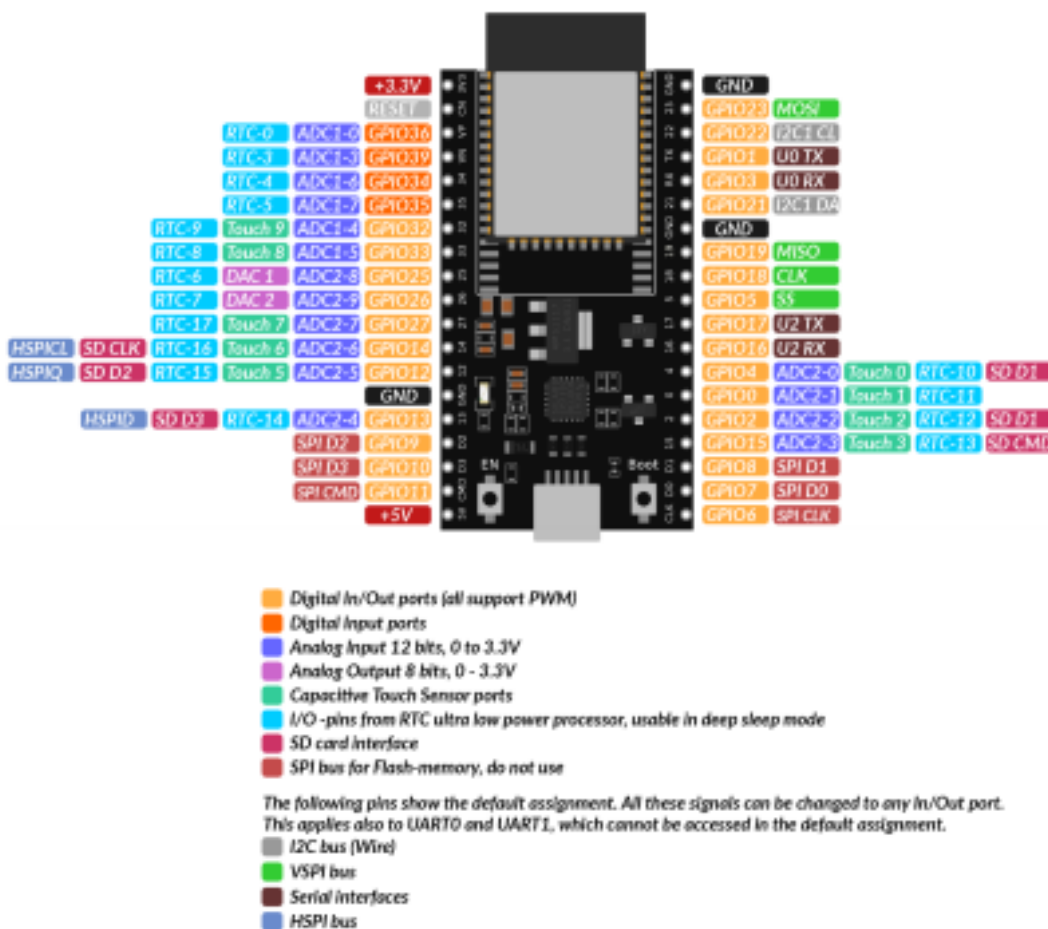


Vendors have consequently created many compact pcb modules based around the ESP32 WROOM-32D chip. Some of these modules have specific identifiers.

# Az-Delivery

## Pinout

The ESP32 Dev Kit C V4 has 38 pins. The pinout is shown on the following image:



For detailed pinout description and I/O capabilities please refer to the datasheet which can be found on the following [link](#).

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



## **Input / Output pins description**

Just like a normal microcontroller board, the ESP32 Dev Kit C V4 has digital input/output pins (GPIO pins - General Purpose Input/Output pins). These digital input/outputs operate at 3.3V.

**5V voltage must not be connected to any ESP32 chip pins!**

The pins are not 5V tolerant, applying more than 3.3V on any pin will destroy the chip.

The GPIO pins 34 to 39 are GPIOs – input only pins. These pins do not have internal pull-ups or pull-down resistors. They cannot be used as outputs, so use these pins only as inputs: GPIO 34, GPIO 35, GPIO 36, GPIO 39

There is an integrated SPI flash on the ESP-WROOM-32 chip. The pins GPIO6 to GPIO 11 are exposed in certain ESP32 development boards. These pins are connected to the integrated SPI flash on the chip and are not recommended for other uses.

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 sensor pins**

The ESP32 has 10 internal capacitive touch sensors. The capacitive touch pins can also be used to wake up the ESP32 from deep sleep. These 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).



### **Analog to Digital converter pins**

The ESP32 has 18 x 12 bits ADC (Analog to Digital converter) input channels (while the ESP8266 only has 1x 10 bits ADC). These are the GPIOs 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).

### **Digital to Analog converter pins**

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

DAC1 (GPIO25), DAC2 (GPIO26).



## **Real Time Clock GPIO pins**

There is RTC (Real time clock) 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 (Pulse Width Modulation) pins**

The ESP32 LED PWM (Pulse width modulation) 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 cannot 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.

## **The I2C interface pins**

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: GPIO 21 (SDA), GPIO 22 (SCL).

## SPI interface pins

By default, the pin mapping for SPI pins is:

SPI	MOSI	MISO	CLK	CS
VSPI		GPIO 19		GPIO 5
HSP I		GPIO 12		GPIO 15

## Strapping Pins

Following pins are used to put the ESP32 into bootloader or flashing mode: GPIO 0, GPIO 2, GPIO 4, GPIO 5 (must be HIGH during boot), GPIO 12 (must be LOW during boot), GPIO 15 (must be HIGH during boot).

Most development boards put the pins in the right state for flashing or boot mode. If some peripherals are connected to the strapping pins and the IDE is unable to upload the code or flash the ESP32, it may be because those peripherals are preventing the ESP32 to enter the right mode. After resetting, flashing, or booting, those pins work as expected. There is Boot Mode Selection documentation guide on the following [link](#). Further and more extensive explanations are not in the scope of this eBook so please, refer to the datasheet.



### **Pins HIGH at Boot**

Some GPIOs change their state to HIGH or output PWM signals at boot or reset. This means that if outputs are connected to these GPIOs this 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 for use), GPIO 14, GPIO 15.

### **Enable (EN)**

Enable (EN) is the 3.3V regulator's enable pin. It has a pulled up state and it needs to be connected to ground to disable the 3.3V regulator. This means that this pin can be connected to a push button to restart your ESP32, for example.



## **USB to Serial communication**

The ESP32 Dev Kit C V4 has a microUSB connection port. It is made around CP21202 chip made by Silicon Laboratories which allows USB to UART serial communication. The chip has the virtual COM port (VCP) feature that appears as COM port in PC applications. The CP2102 UART interface implements all RS-232 signals, including control and handshaking signals, so existing system firmware does not need to be modified. To be able to use the ESP32 the driver has to be installed.



## WiFi Communication

ESP32 Dev Kit C V4 has integrated Wi-Fi communication interface and can operate in three different modes: Wi-Fi station, Wi-Fi access point, and both at the same time. It supports the following features:

- 802.11b and 802.11g data-rates
- 802.11n MCS0-7 in both 20MHz and 40MHz bandwidth
- 802.11n MCS32
- 802.11n 0.4 $\mu$ S guard-interval
- Data-rate up to 150 Mbps
- Receiving STBC 2x1
- Up to 20 dBm transmitting power
- Adjustable transmitting power
- Antenna diversity and selection (software-managed hardware)





## **Bluetooth Communication**

The ESP32 Dev Kit C V4 has an integrated Bluetooth Radio and supports following features:

- Class-1, class-2 and class-3 transmit output powers and over 30 dB dynamic control range
- $\pi/4$  DQPSK and 8 DPSK modulation
- High performance in NZIF receiver sensitivity with over 98 dB dynamic range
- Class-1 operation without external PA
- Internal SRAM allows full speed data transfer, mixed voice and data, and full piconet operation
- Logic for forward error correction, header error control, access code correlation, CRC, demodulation, encryption bit stream generation, whitening and transmit pulse shaping
- ACL, SCO, eSCO and AFH
- A-law,  $\mu$ -law and CVSD digital audio CODEC in PCM interface
- SBC audio CODEC
- Power management for low power applications
- SMP with 128-bit AES



Also, the Bluetooth Radio has support for the following communication interface protocols:

- UART HCI interface, up to 4 Mbps
- SDIO / SPI HCI interface
- I2C interface
- PCM / I2S audio interface.



### **Other features**

ESP32-WROOM 32D chip has an integrated Hall Effect Sensor that detects changes in the magnetic field in its surroundings.

The Hall sensor is based on an N-carrier resistor. When the chip is in the magnetic field, the Hall sensor develops a small voltage on the resistor, which can be directly measured by the analog-digital converter (ADC), or amplified by the ultra low noise analog pre-amplifier and then measured by the ADC.


The temperature sensor generates a voltage that varies with temperature. The voltage is internally converted via an analog-to-digital converter into a digital code. The temperature sensor has a range of  $-40^{\circ}\text{C}$  to  $125^{\circ}\text{C}$ . As the offset of the temperature sensor varies from chip to chip due to process variation, together with the heat generated by the Wi-Fi circuitry itself (which affects measurements), the internal temperature sensor is only suitable for applications that detect temperature changes instead of absolute temperatures and for calibration purposes as well. However, if the user calibrates the temperature sensor and uses the device in a minimally powered-on application, the results could be accurate enough.

# Az-Delivery

## How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the [link](#) and download the installation file for the operating system of choice. The Arduino IDE version used for this eBook is **1.8.13**.

### Download the Arduino IDE




#### ARDUINO 1.8.13

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

**Windows** Installer, for Windows 7 and up  
**Windows** ZIP file for non admin install

**Windows app** Requires Win 8.1 or 10 

**Mac OS X** 10.10 or newer

**Linux** 32 bits  
**Linux** 64 bits  
**Linux** ARM 32 bits  
**Linux** ARM 64 bits

[Release Notes](#)  
[Source Code](#)  
[Checksums \(sha512\)](#)

For *Windows* users, double click on the downloaded .exe file and follow the instructions in the installation window.

# Az-Delivery

For *Linux* users, download a file with the extension *.tar.xz*, which has to be extracted. When it is extracted, go to the extracted directory and open the terminal in that directory. Two *.sh* scripts have to be executed, the first called *arduino-linux-setup.sh* and the second called *install.sh*.

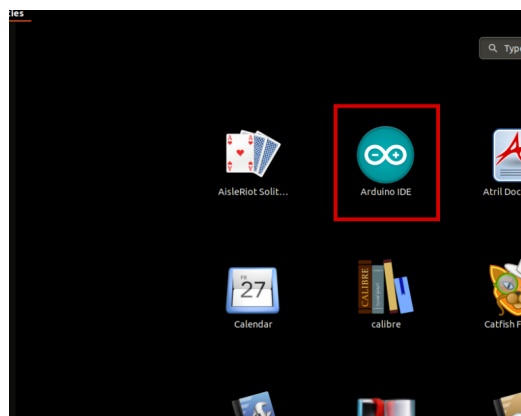
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

**sh arduino-linux-setup.sh user\_name**

**user\_name** - is the name of a superuser in the Linux operating system. A password for the superuser has to be entered when the command is started. Wait for a few minutes for the script to complete everything.

The second script, called *install.sh*, has to be used after the installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

After the installation of these scripts, go to the *All Apps*, where the *Arduino IDE* is installed.



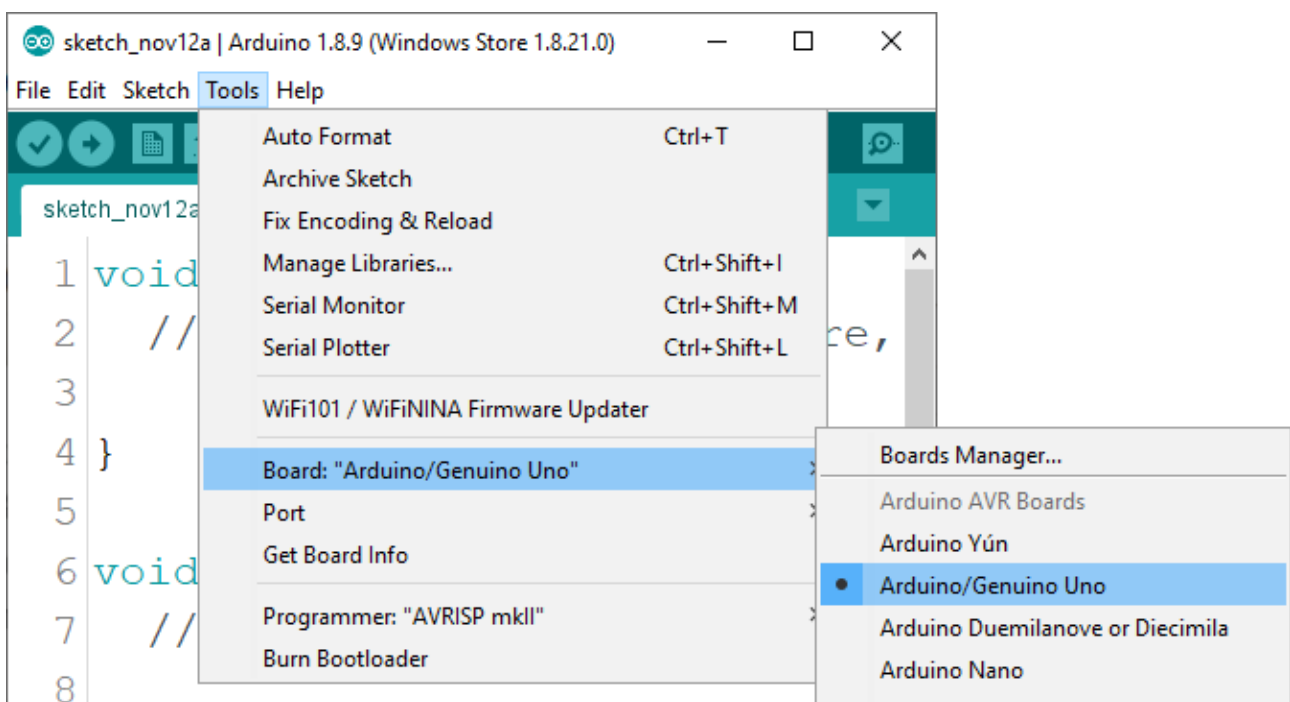
# Az-Delivery

Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check, if your PC can detect an microcontroller board. Open freshly installed Arduino-IDE, and go to:

*Tools > Board > {your board name here}*

*{your board name here}* should be the *Arduino/Genuino Uno*, as it can be seen on the following image:



The port to which the microcontroller board is connected has to be selected.

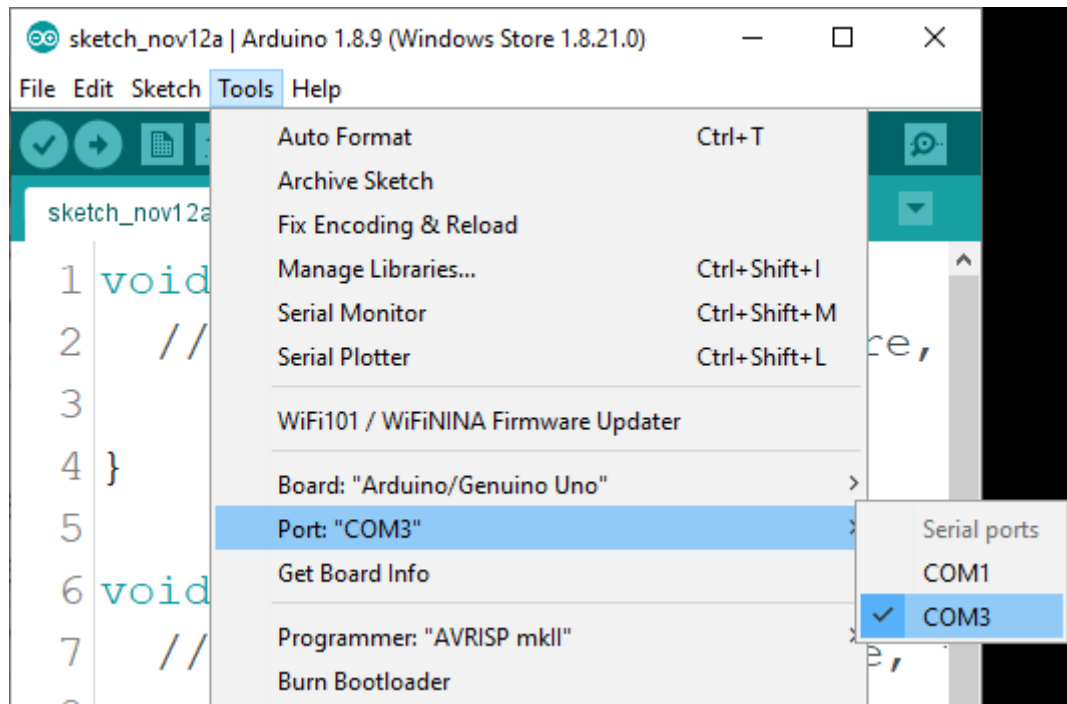
Go to: *Tools > Port > {port name goes here}*

and when the microcontroller board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

# Az-Delivery

If the Arduino IDE is used on Windows, port names are as

follows:



For *Linux* users, for example port name is `/dev/ttyUSBx`, where *x* represents integer number between 0 and 9.



### **Additional setup**

In order to use ESP32 Dev Kit C V4 with Arduino IDE, follow few easy steps. Before setting the Arduino IDE, the driver for the USB to Serial communication has to be installed. If the driver is not installed automatically, there is a support page that contains the drivers for Windows/Mac or Linux and can be chosen depending on which one is used. Drivers can be downloaded from the following [link](#).

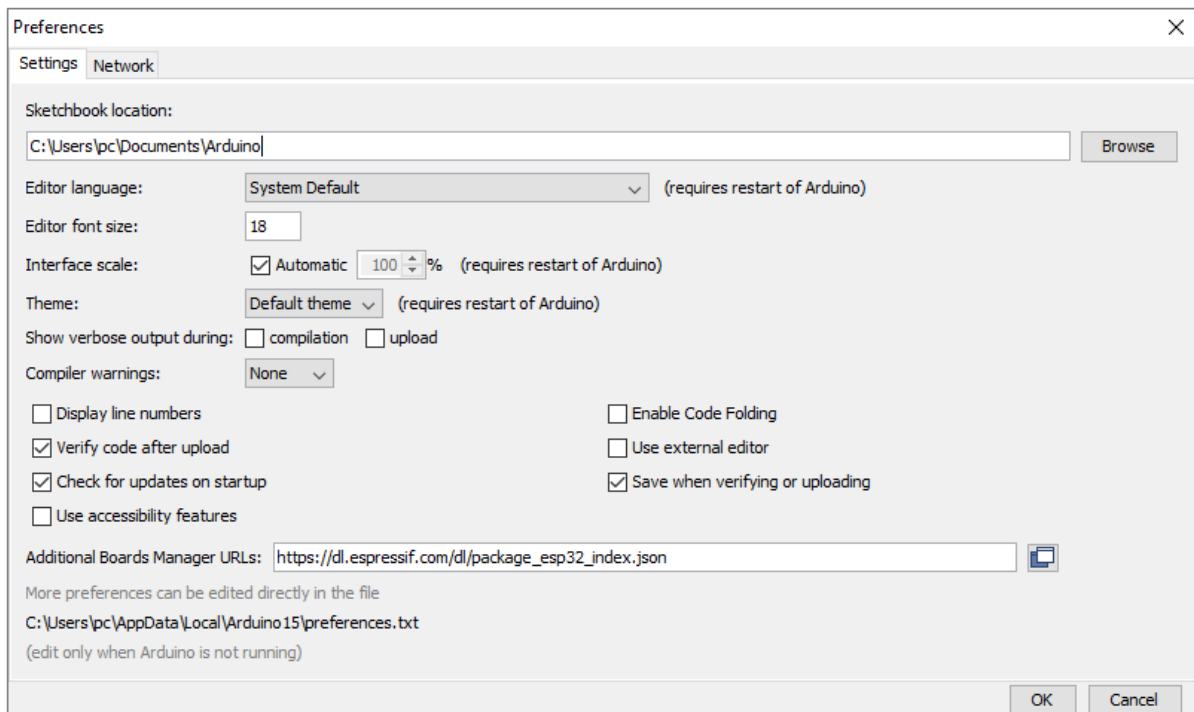


# Az-Delivery

Next, to install support for the ESP32 platform, open Arduino IDE and go to: *File > Preferences*, and find Additional URLs field.

Then copy the following URL:

[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)



# Az-Delivery

Paste this link in the Additional URLs field. If one or more links are inside this field, just add one comma after the last link, paste new link after the comma and click the *OK* button.



Open Arduino IDE again and go to:

*Tools > Board > Boards Manager*

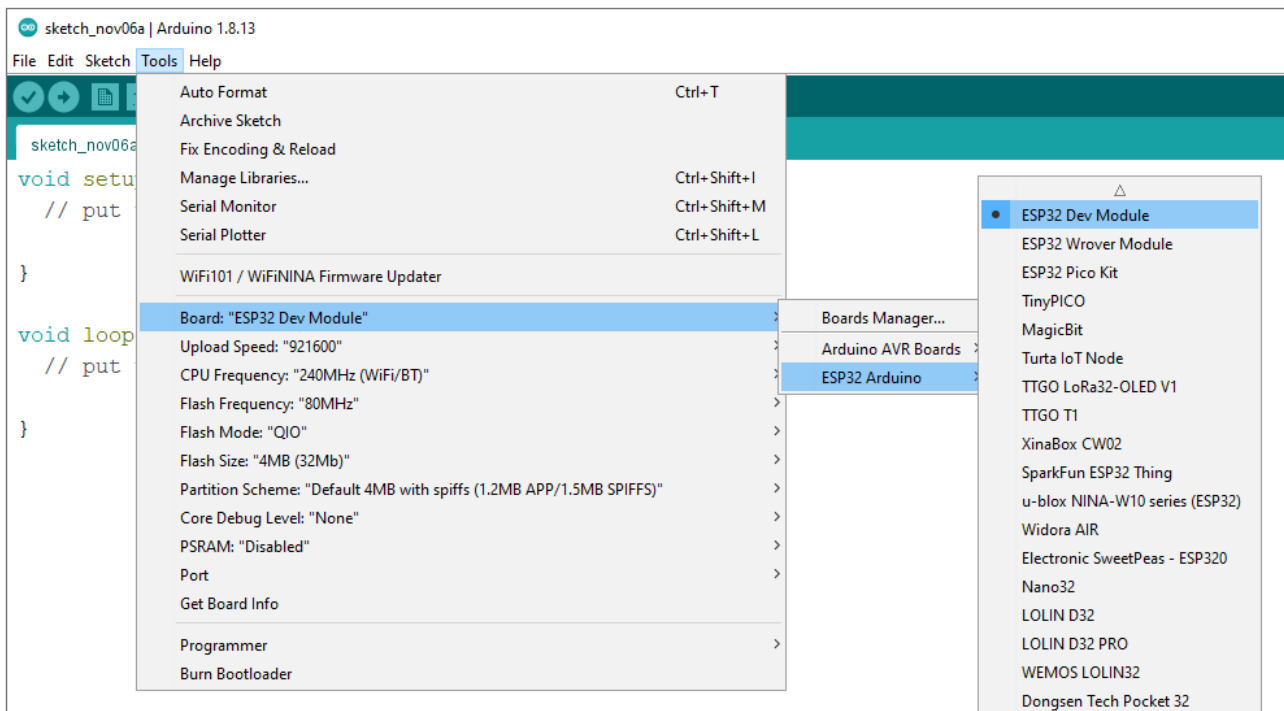
When new window opens, type *esp32* in the search box and install the board called *esp32* made by *Espressif Systems*, as shown on the following image:



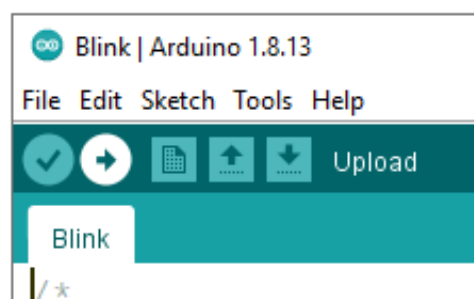
# Az-Delivery

To select ESP32 board, go to:

*Tools > Board > ESP32 > ESP32 Dev Module*



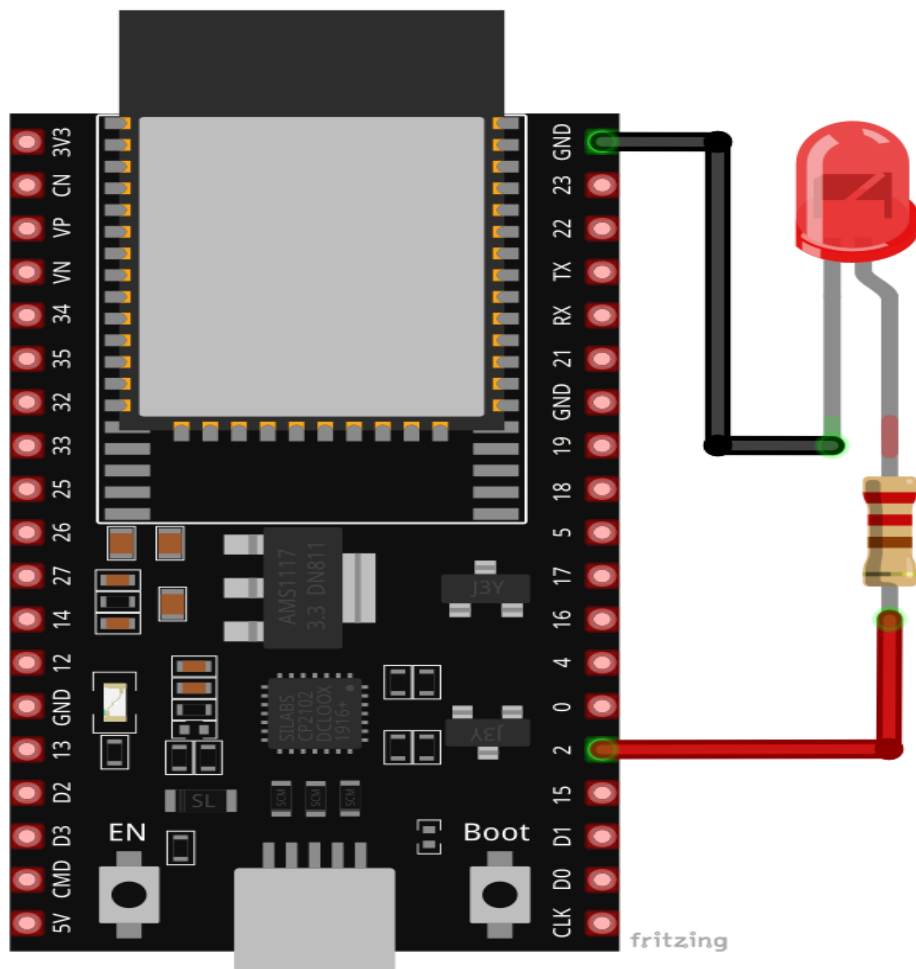
To upload the sketch code to the ESP32 board, first select port on which you connected the board. Go to: *Tools > Port > {port name}0*



# Az-Delivery

## ESP32 Dev Kit C V4 wiring example

Connect the ESP32 Dev Kit C V4 with an LED and resistor as shown on the following connection diagram:



ESP32 Dev Kit C V4 pin	LED pin	Wire color
GPIO2 (pin2)	Anode (+) through resistor	Red wire
GND	Cathode (-)	Black wire

# Az-Delivery

## Sketch examples

### Blinking LED

```
int ledPin = 2;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```



## PWM - Pulse Width Modulation

```
#define LEDC_CHANNEL_0 0
#define LEDC_TIMER_13_BIT 13
#define LEDC_BASE_FREQ 5000
#define LED_PIN 2

int brightness = 0;
int fadeAmount = 5;

void ledcAnalogWrite(uint8_t channel, uint32_t value, uint32_t valueMax = 255) {
    uint32_t duty = (8191 / valueMax) * min(value, valueMax); ledcWrite(channel,
duty);
}

void setup() {
    ledcSetup(LEDC_CHANNEL_0, LEDC_BASE_FREQ, LEDC_TIMER_13_BIT);
    ledcAttachPin(LED_PIN, LEDC_CHANNEL_0);
}

void loop() {
    ledcAnalogWrite(LEDC_CHANNEL_0, brightness);
    brightness = brightness + fadeAmount;
    if (brightness <= 0 || brightness >= 255) {
        fadeAmount = -fadeAmount;
    }
    delay(30);
}
```



Now it is the time to learn and make your own projects. You can do that with the help of many example scripts and other tutorials, which can be found on the Internet.

**If you are looking for the high quality microelectronics and accessories, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.**

<https://az-delivery.de>

**Have Fun!**

**Impressum**

<https://az-delivery.de/pages/about-us>