

# **Preface**

# Content

*Chapter*

*Page No.*

## **1. Arduino ..... 1**

1.1 *Introduction.....*

1.2 *Board Type.....*

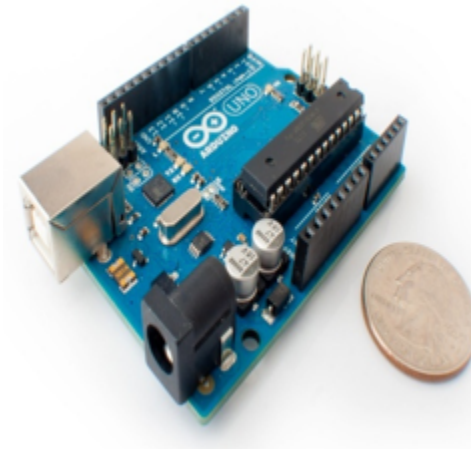
1.3 *Installation.....*

1.4 *Programme Structure.....*

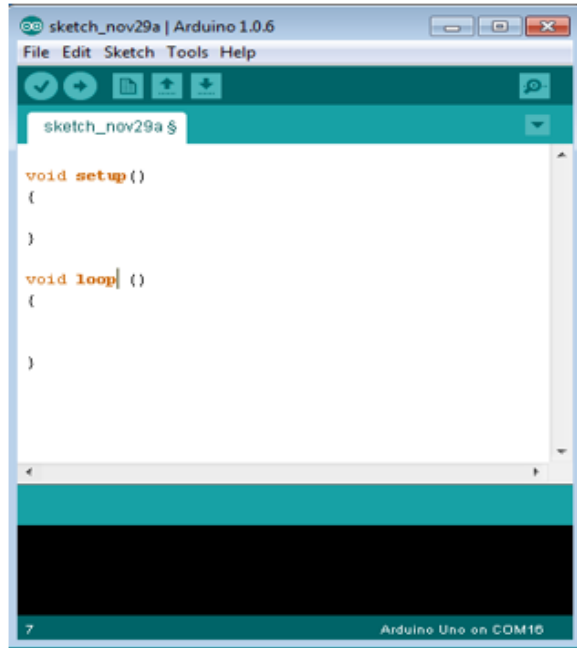
***Workshop Time.....***

# Chapter – 1

## Arduino



(Arduino UNO Board)



(Arduino IDE)

### 1.1 Introduction

Arduino is a prototype platform (open-source) based on easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

The key features are –

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

## 1.2 Board Types

Various kinds of Arduino boards are available depending on different microcontrollers used. However, all Arduino boards have one thing in common: they are programmed through the Arduino IDE.

The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor etc. Some boards are designed to be embedded and have no programming interface (hardware), which you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V. Here is a list of different Arduino boards available.

### Arduino boards based on ATMEGA328 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Uno R3	5V	16M Hz	14	6	6	1	USB via ATmega16U2
Arduino Pro 3.3v/8 MHz	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
Arduino Pro mini 3.3v/8mhz	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Ethernet	5V	16M Hz	14	6	6	1	FTDI-Compatible Header
LilyPad Arduino 328 main board	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
LilyPad Arduino simple board	3.3V	8MHz	9	4	5	0	FTDI-Compatible Header

#### Arduino boards based on ATMEGA32u4 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Leonardo	5V	16MHz	20	12	7	1	Native USB
Pro micro 3.3V/8MHz	5V	16MHz	14	6	6	1	Native USB
LilyPad Arduino USB	3.3V	8MHz	14	6	6	1	Native USB

### Arduino boards based on ATMEGA2560 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	5V	16MHz	54	16	14	4	USB via ATmega16U2B
Mega Pro 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header

### Arduino boards based on AT91SAM3X8E microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	3.3V	84MHz	54	12	12	4	USB native

## 1.3 Installation

---

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board. In this section, we will learn in easy steps how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

**Step 1** – First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.



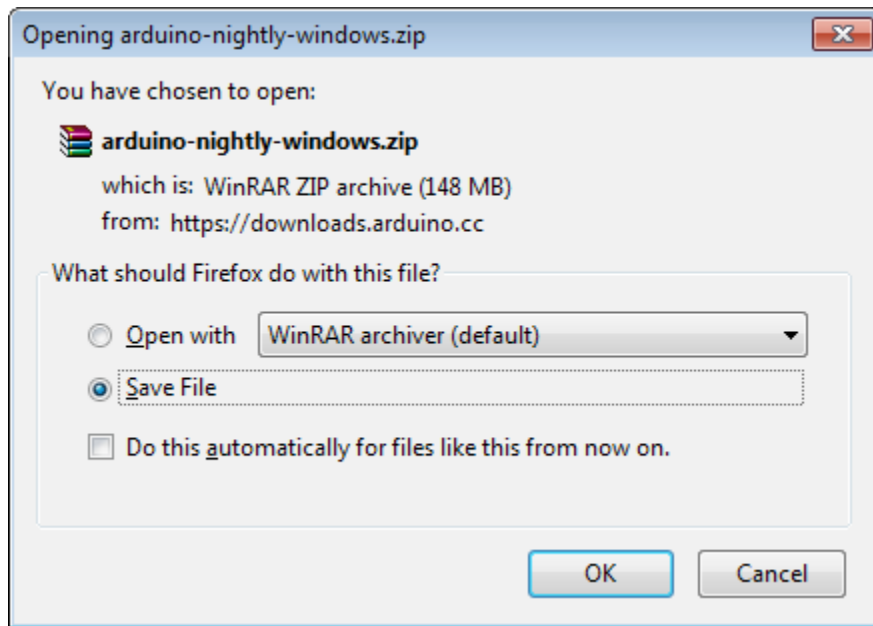
In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.





## Step 2 – Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



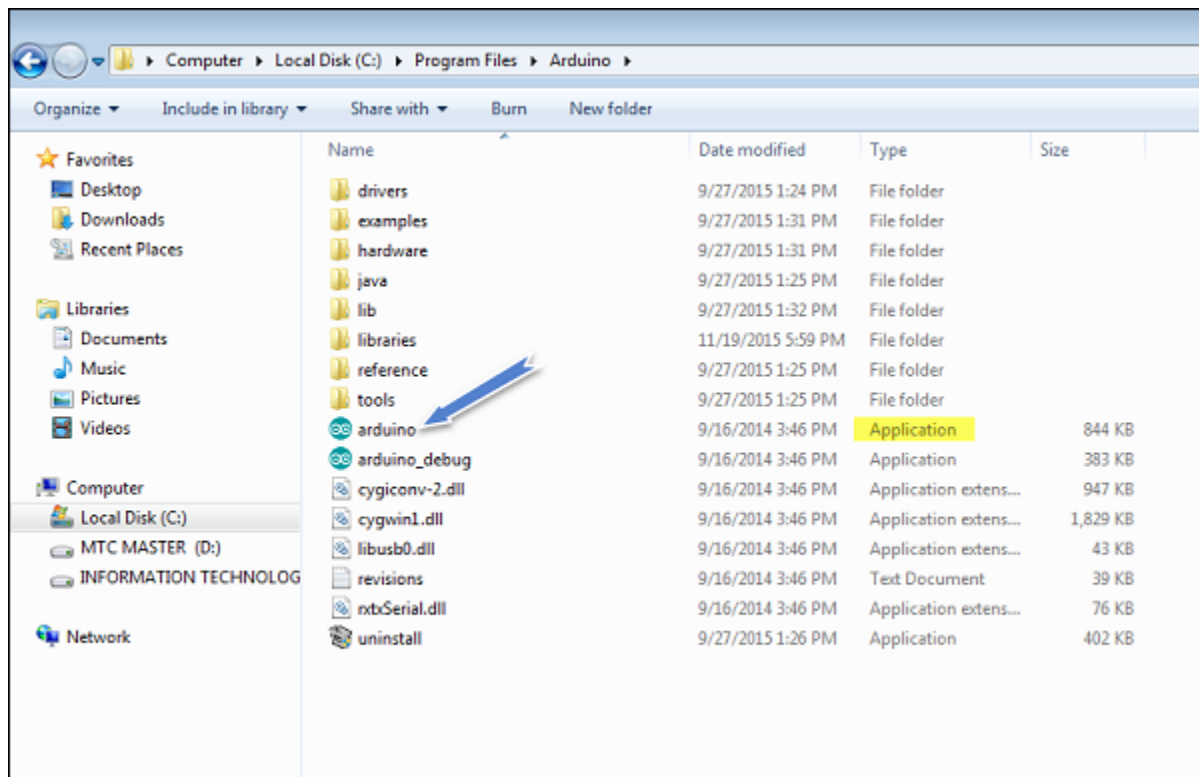
## Step 3 – Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

#### Step 4 – Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

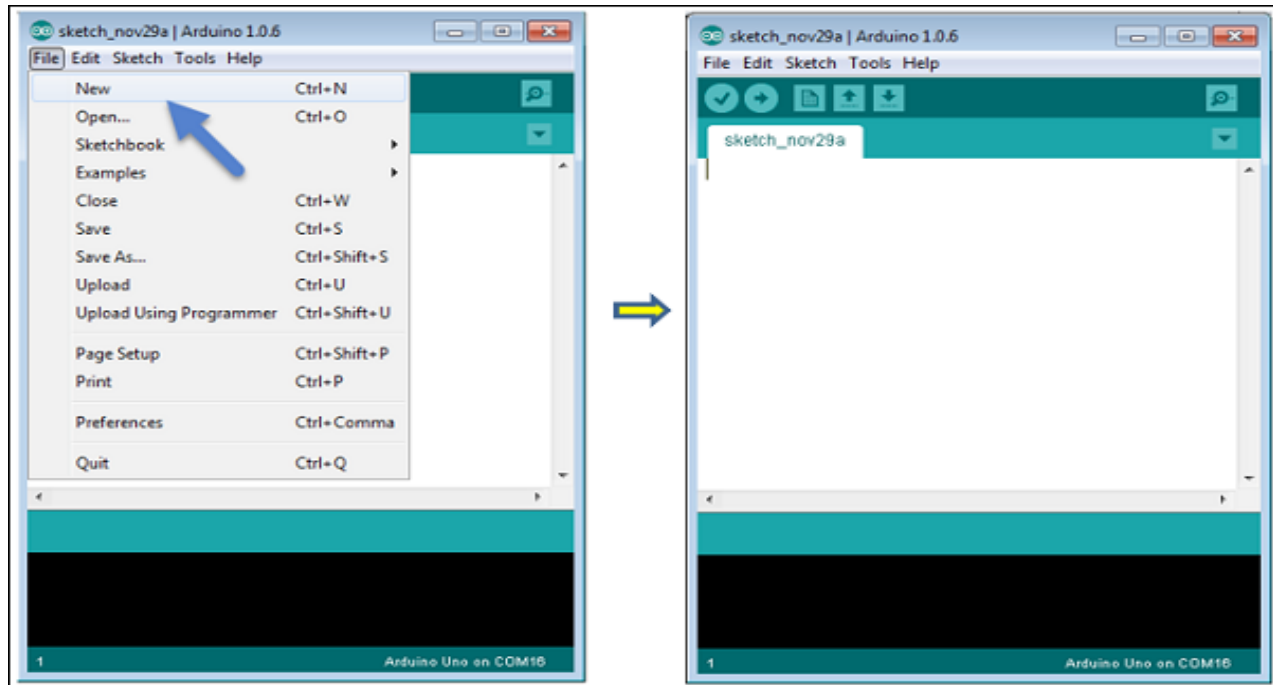


#### Step 5 – Open your first project.

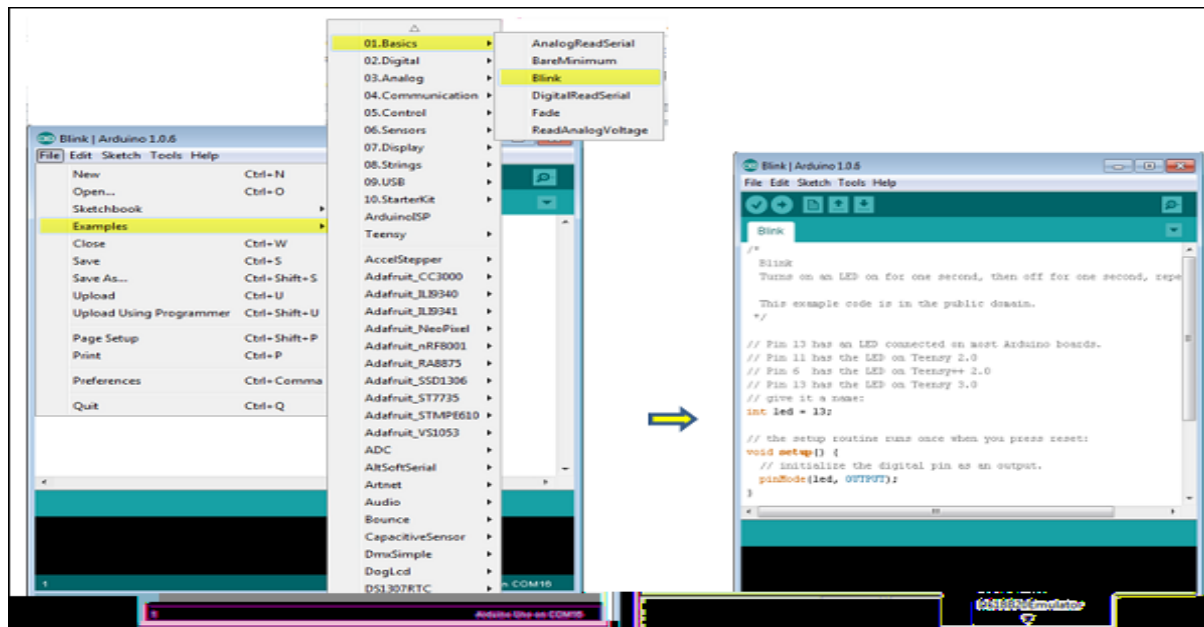
Once the software starts, you have two options –

- Create a new project.
- Open an existing project example.

To create a new project, select File → New.



To open an existing project example, select File → Example → Basics → Blink.

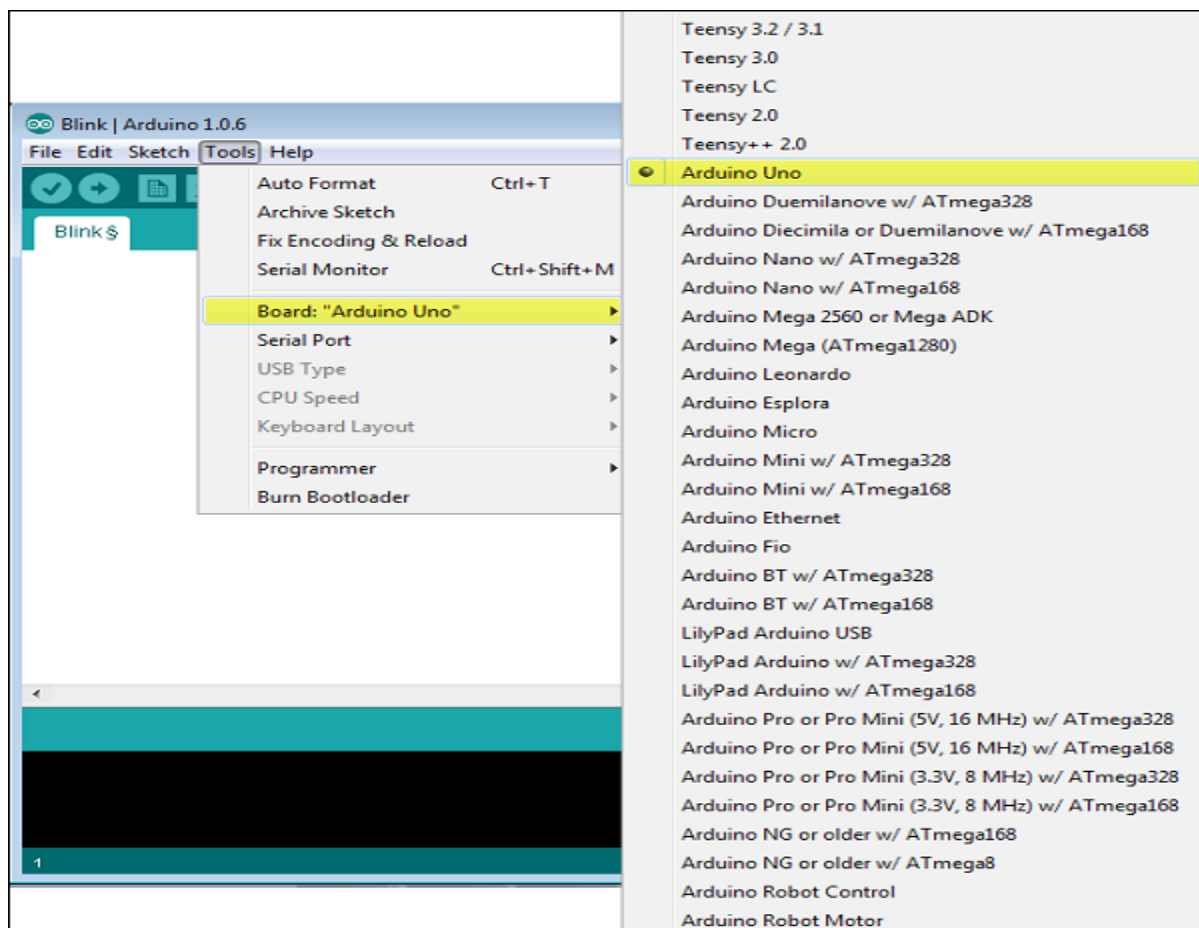


Here, we are selecting just one of the examples with the name '**Blink**'. It turns the LED on and off with some time delay. You can select any other example from the list.

## Step 6 – Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

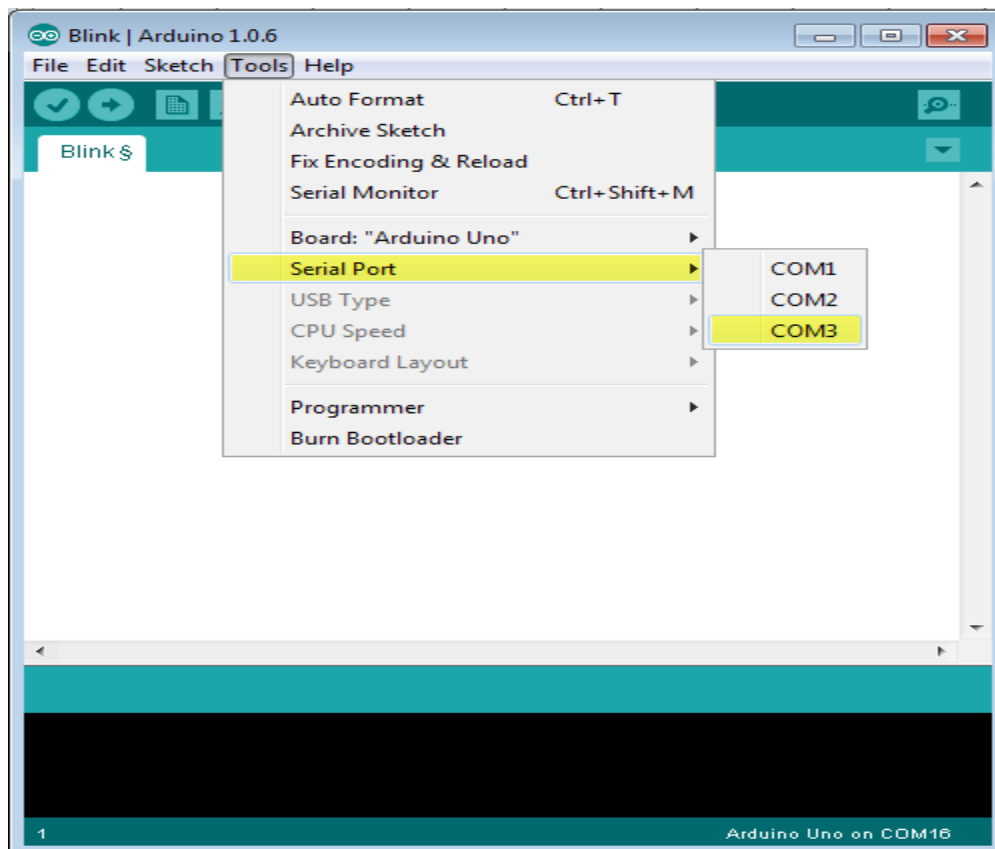
**Go to Tools** → Board and select your board.



Here, we have selected the Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

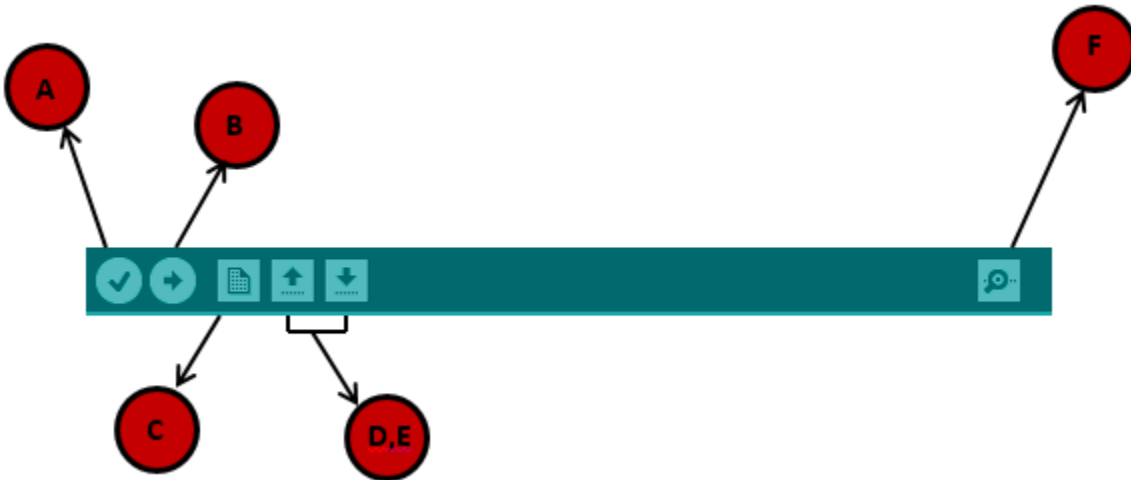
### Step 7 – Select your serial port.

Select the serial device of the Arduino board. Go to **Tools** → **Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



## Step 8 – Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



**A**– Used to check if there is any compilation error.

**B**– Used to upload a program to the Arduino board.

**C**– Shortcut used to create a new sketch.

**D**– Used to directly open one of the example sketches.

**E**– Used to save your sketch.

**F**– Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

**Note** – If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

## 1.4 Programme Structure

In this chapter, we will study in depth the Arduino program structure and we will learn more new terminologies used in the Arduino world. The Arduino software is open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.

**Sketch** – The first new terminology is the Arduino program called “**sketch**”.

### Structure

Arduino programs can be divided into three main parts: **Structure**, **Values**(variables and constants), and **Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the **Structure**. Software structure consists of two main functions –

- Setup( ) function
- Loop( ) function

```
void setup ( ) {  
  
}
```

The **setup()** function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

```
void Loop ( ) {  
  
}
```

After creating a **setup()** function, which initializes and sets the initial values, the **loop()** function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

## 1.5 Sensors

### What are the sensors?

- The sensors are defined as a machine, module, or a device that detects changes in the environment. The sensors transfer those changes to the electronic devices in the form of a signal.
- A sensor and electronic devices always work together. The output signal is easily readable by humans.
- Nowadays, Sensors are used in daily lives. For example, controlling the brightness of the lamp by touching its base, etc. The use of sensors is expanding with new technologies.

### How are the sensors used in Arduino?

The data signal runs from the sensor to the output pins of the **Arduino**. The data is further recorded by the Arduino.



Some of the types of sensors in Arduino are listed below:

- **Light sensor**



The light sensor is used to control the light. It is used with LDR (Light Dependent Resistor) in Arduino.

- **Ultrasonic sensor**



The ultrasonic sensor is used to determine the distance of the object using SONAR.

- **Temperature sensor**



The temperature sensor is used to detect the temperature around it.

- **Knock Sensor**

A knock sensor is essentially a small “listening” device in or on the engine that detects these irregular vibrations and sounds that come from the engine block.

The knock sensor picks up vibration and sound coming from the engine block, turns it into an electronic signal and sends that signal to the engine control unit (ECU). The car’s computer then judges the information and determines whether or not ignition timing should be altered.

- **Object Detection Sensor**



It is used to detect the object by emitting infrared radiations, which are reflected or bounced back by that object.

- **Tracking Sensor**

It allows the robots to follow a particular path specified by sensing the marking or lines on the surface.

- **Metal Touch Sensor**

It is suitable for detecting the human touch.

- **Water Level Sensor**

It is used to measure the water or the depth of the water level. It is also used to detect leaks in containers.

- **Vibration Sensor**

The vibration sensor is used to measure the vibrations.

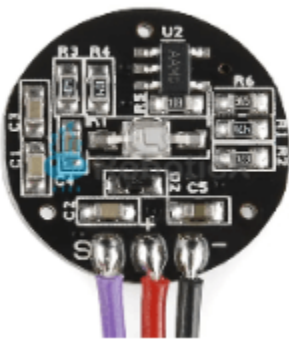
- **Air Pressure sensor**

It is commonly related to meteorology, biomedical fields. It looks like the below image:



- **Pulse Sensor**

The pulse sensor is used to measure the pulse rate. It looks like the below image:



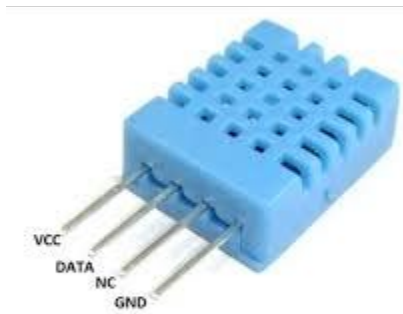
- **Capacitive soil moisture sensor**

It is used to measure the moisture level of the soil.

- **Microphone sensor**

The microphone sensor in Arduino is used to detect the sound. The analog and digital are the two outputs of this module. The digital output sends the high signal when the intensity of sound reaches a certain threshold. We can adjust the sensitivity of a module with the help of a potentiometer.

- **humidity sensor**



The humidity sensor is used to monitor weather conditions.

- **Motion sensor**

The motion sensor detects the movement and occupancy from the human body with the help of Infrared radiation.

- **Vibration sensor**

The vibration sensor is used to detect the vibrations.

- **Sound sensor**

The sound sensor is suitable to detect the sound of the environment.

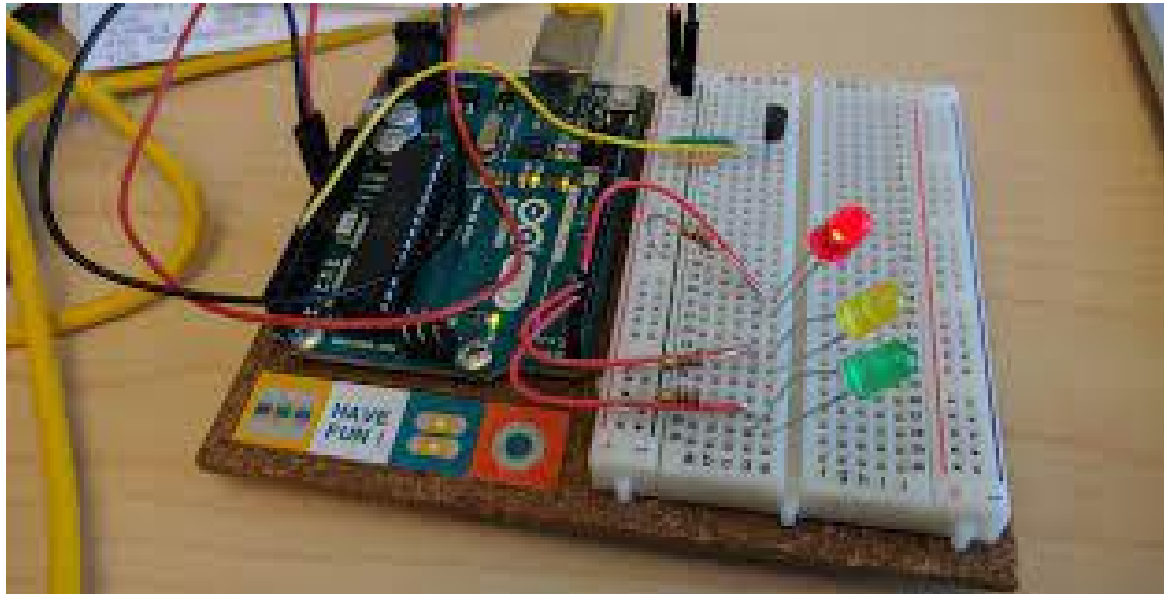
- **Pressure Sensor**

The pressure sensor is used to measure the pressure. The sensor in Arduino measures the pressure and displays it on the small LCD screen.

- **Magnetic field sensor**

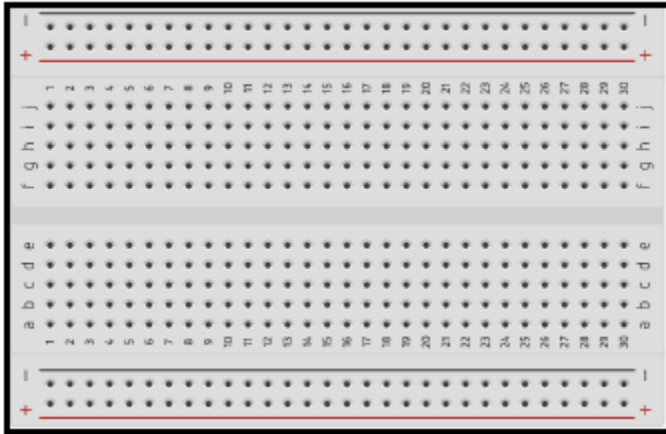
The magnetic field sensor measures the magnetic field strength and produces a varying voltage as the output in Arduino.

## 1.6 Workshop Time



# Tools Required

## 1. Breadboard



## 2. Jumper / Connecting Wires





# Chapter – 2

## ESP32

**2.1 Introduction.** The ESP32 is a series of chip microcontrollers . 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.

- Low-power: the ESP32 consumes very little power compared with other microcontrollers, and it supports low-power mode states like [deep sleep](#) to save power.
- [Wi-Fi capabilities](#): the ESP32 can easily connect to a Wi-Fi network to connect to the internet (station mode), or create its own Wi-Fi wireless network ([access point mode](#)) so other devices can connect to it—this is essential for IoT and Home Automation projects—you can have multiple devices communicating with each other using their Wi-Fi capabilities;
- Bluetooth: the ESP32 supports [Bluetooth classic](#) and [Bluetooth Low Energy \(BLE\)](#)—which is useful for a wide variety of IoT applications;
- [Dual-core](#): most ESP32 are dual-core
- [Rich peripheral input/output interface](#)—the ESP32 supports a wide variety of input (read data from the outside world) and output (to send commands / signals to the outside world) peripherals like [capacitive touch](#), [ADCs](#), [DACs](#), [UART](#), [SPI](#), [I2C](#), [PWM](#), and much more.

- [Compatible with the Arduino “programming language”](#): those that are already familiar with programming the Arduino board, you’ll be happy to know that they can program the ESP32 in the Arduino style.
- [Compatible with MicroPython](#): you can program the ESP32 with MicroPython firmware, which is a re-implementation of Python 3 targeted for microcontrollers and embedded systems.

## 2.2 Specifications.

If you want to get a bit more technical and specific, you can take a look at the following detailed specifications of the ESP32 (source: <http://esp32.net/>)—for more details, [check the datasheet](#))

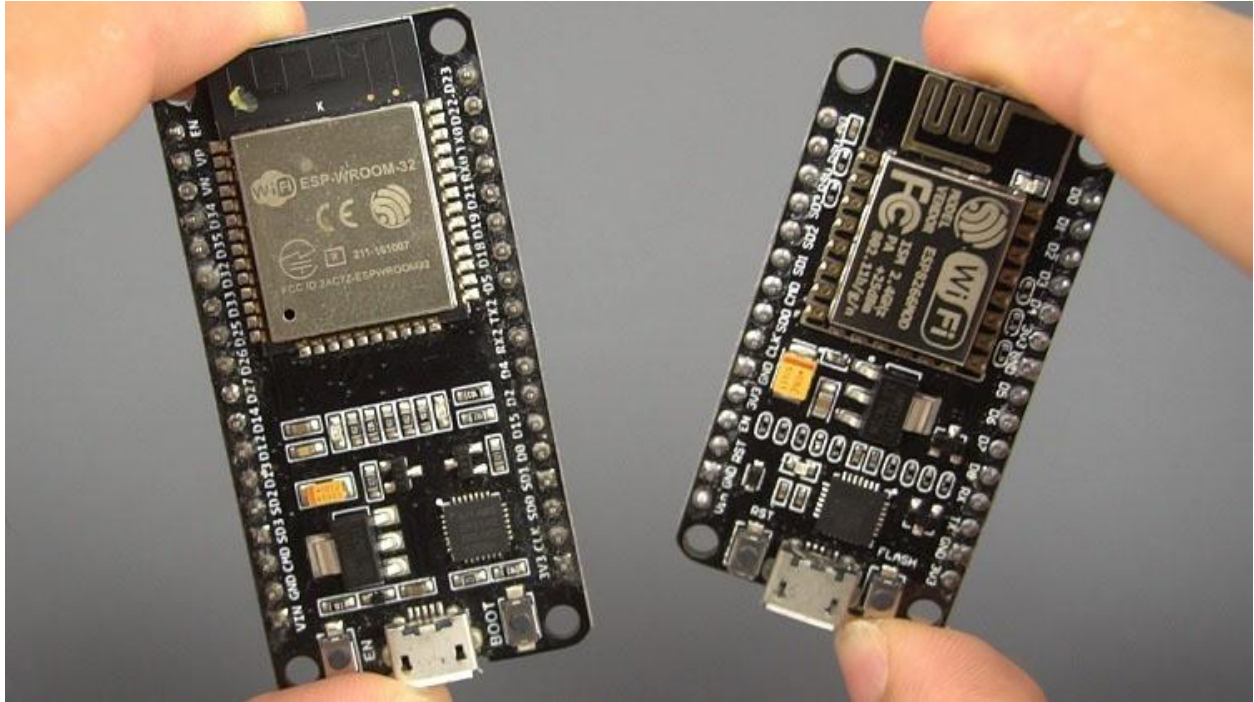


ESP32 module: ESP-WROOM-32

- Wireless connectivity [WiFi](#): 150.0 Mbps data rate with HT40
  1. Bluetooth: [BLE \(Bluetooth Low Energy\)](#) and [Bluetooth Classic](#)
- Processor: Tensilica Xtensa Dual-Core 32-bit LX6 microprocessor, running at 160 or 240 MHz
- Memory:

- ROM: 448 KB (for booting and core functions)
- SRAM: 520 KB (for data and instructions)
- RTC fast SRAM: 8 KB (for data storage and main CPU during RTC Boot from the deep-sleep mode)
- RTC slow SRAM: 8KB (for co-processor accessing during deep-sleep mode)
- eFuse: 1 Kbit (of which 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including Flash-Encryption and Chip-ID)
- Embedded flash: flash connected internally via IO16, IO17, SD\_CMD, SD\_CLK, SD\_DATA\_0 and SD\_DATA\_1 on ESP32-D2WD and ESP32-PICO-D4.
  - 0 MiB (ESP32-D0WDQ6, ESP32-D0WD, and ESP32-S0WD chips)
  - 2 MiB (ESP32-D2WD chip)
  - 4 MiB (ESP32-PICO-D4 SiP module)
- Low Power: ensures that you can still use ADC conversions, for example, during [deep sleep](#).
- [Peripheral Input/Output](#):
  - peripheral interface with DMA that includes [capacitive touch](#)
  - [ADCs \(Analog-to-Digital Converter\)](#)
  - DACs (Digital-to-Analog Converter)
  - [I<sup>2</sup>C \(Inter-Integrated Circuit\)](#)
  - UART (Universal Asynchronous Receiver/Transmitter)
  - [SPI \(Serial Peripheral Interface\)](#)
  - I<sup>2</sup>S (Integrated Interchip Sound)
  - RMI (Reduced Media-Independent Interface)
  - [PWM \(Pulse-Width Modulation\)](#)
- Security: hardware accelerators for AES and SSL/TLS

## 2.3 Main Differences Between ESP32 and ESP8266



Previously, we mentioned that the ESP32 is the ESP8266 successor. What are the main differences between ESP32 and ESP8266 boards?

The ESP32 adds an [extra CPU core](#), faster [Wi-Fi](#), [more GPIOs](#), and supports [Bluetooth 4.2](#) and [Bluetooth low energy](#). Additionally, the ESP32 comes with [touch-sensitive pins](#) that can be used to [wake up the ESP32 from deep sleep](#), and [built-in hall effect sensor](#).

Both boards are cheap, but the ESP32 costs slightly more. While the ESP32 can cost around \$6 to \$12, the ESP8266 can cost \$4 to \$6 (but it really depends on where you get them and what model you're buying).

So, in summary:

- The ESP32 is faster than the ESP8266;

- The ESP32 comes with more GPIOs with multiple functions;
- The ESP32 supports analog measurements on 18 channels (analog-enabled pins) versus just one 10-bit ADC pin on the ESP8266;
- The ESP32 supports Bluetooth while the ESP8266 doesn't;
- The ESP32 is dual-core (most models), and the ESP8266 is single core;
- The ESP32 is a bit more expensive than the ESP8266.

For a more detailed analysis of the differences between those boards, we recommend reading the following article: [ESP32 vs ESP8266 – Pros and Cons](#).

## 2.4 **ESP32 Development Boards**

ESP32 refers to the bare ESP32 chip. However, the “ESP32” term is also used to refer to ESP32 development boards. Using ESP32 bare chips is not easy or practical, especially when learning, testing, and prototyping. Most of the time, you’ll want to use an ESP32 development board.





These development boards come with all the needed circuitry to power and program the chip, connect it to your computer, pins to connect peripherals, built-in power and control LEDs, an antenna for wi-fi signal, and other useful features. Others even come with extra hardware like specific sensors or modules, displays, or a camera in the case of the ESP32-CAM.

## 2.5 What is the best ESP32 development board for beginners?

For beginners, we recommend an ESP32 board with a vast selection of available GPIOs, and without any extra hardware features. It's also important that it comes with voltage regular and USB input for power and upload code.

In most of our ESP32 projects, we use the [ESP32 DEVKIT DOIT board](#), and that's the one we recommend for beginners. There are different versions of this board with a different number of available pins (30, 36, and 38)—all boards work in a similar way.



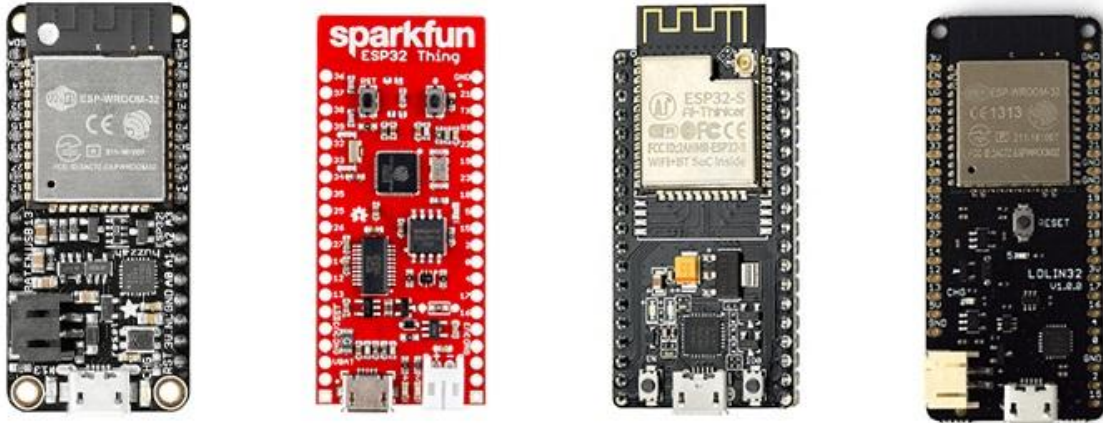
### Where to Buy?

You can check the following link to find the ESP32 DEVKIT DOIT board in different stores:

- [ESP32 DEVKIT DOIT board](#)

Other similar boards with the features mentioned previously may also be a good option like the Adafruit ESP32 Feather, Sparkfun ESP32 Thing, NodeMCU-32S, Wemos LoLin32, etc.





## 2.6 ESP32 DEVKIT DOIT

The picture below shows the ESP32 DEVKIT DOIT V1 board, version with 36 GPIO pins.



## Specifications – ESP32 DEVKIT V1 DOIT

The following table shows a summary of the ESP32 DEVKIT V1 DOIT board features and specifications:

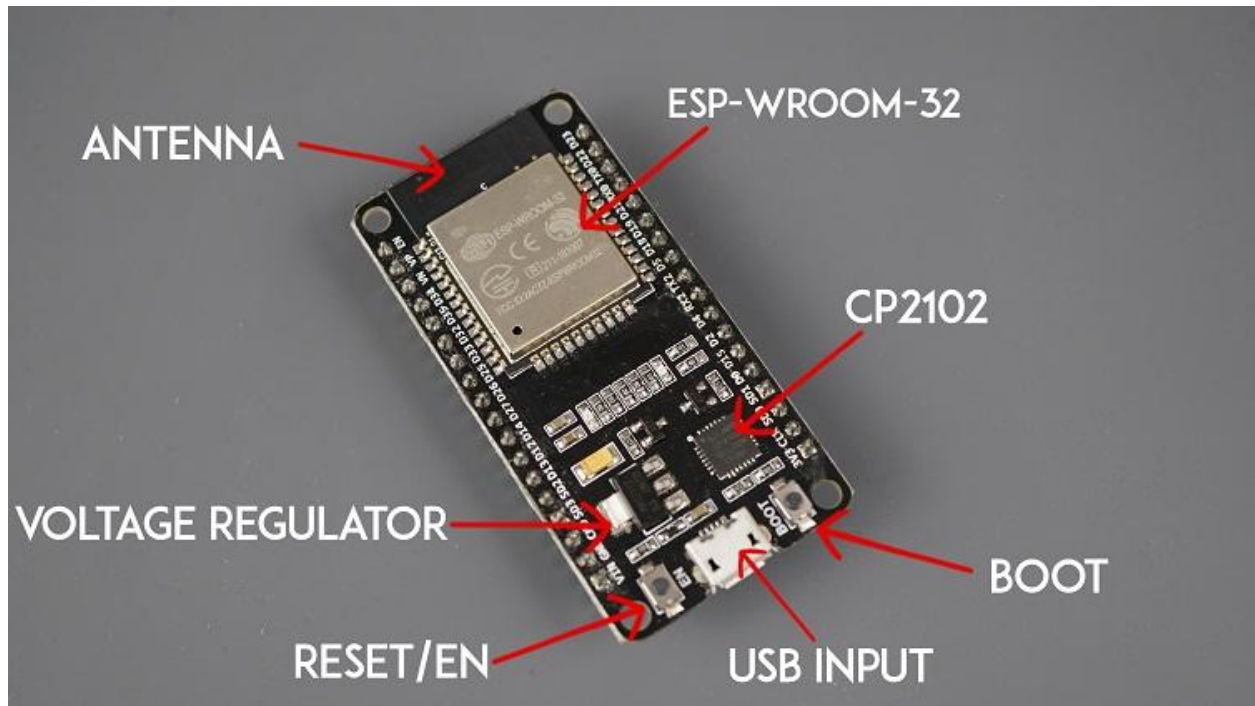


Number of cores	2 (dual core)
Wi-Fi	2.4 GHz up to 150 Mbits/s
Bluetooth	BLE (Bluetooth Low Energy) and legacy Bluetooth
Architecture	32 bits
Clock frequency	Up to 240 MHz
RAM	512 KB
Pins	30, 36, or 38 (depending on the model)
Peripherals	Capacitive touch, ADC (analog to digital converter), DAC (digital to analog converter), I2C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I2S (Integrated Inter-IC Sound), RMII (Reduced

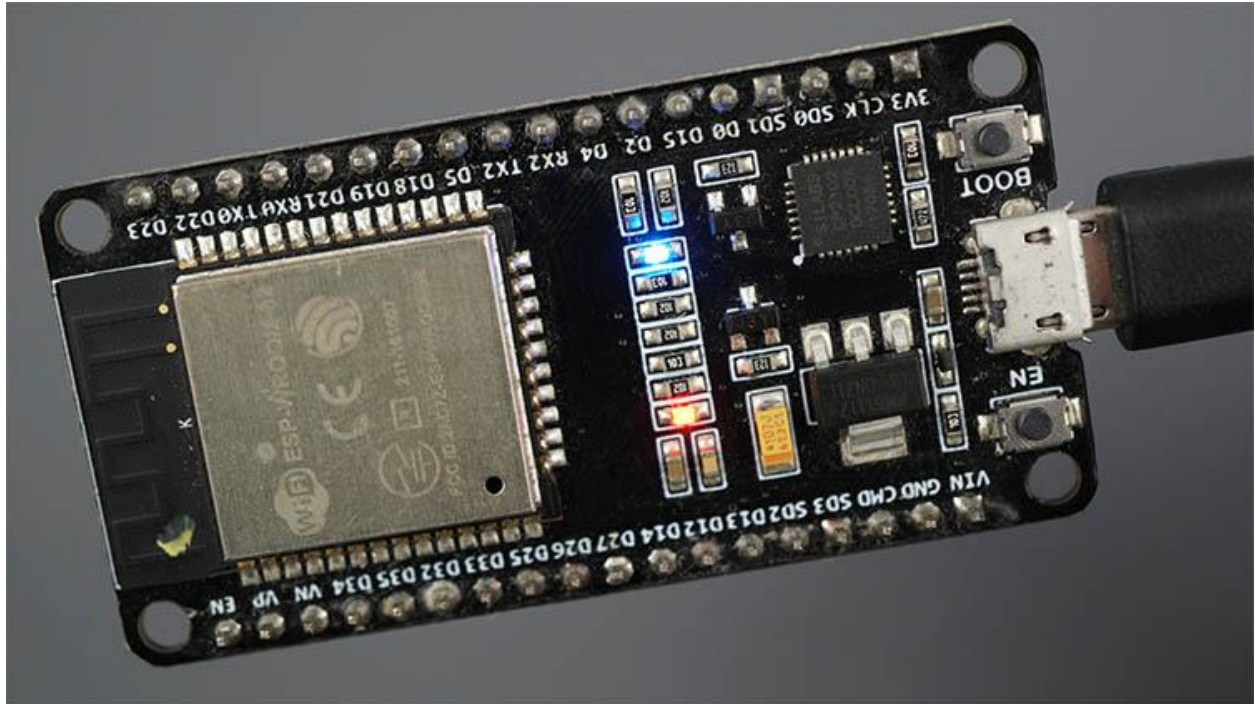
	Media-Independent Interface), PWM (pulse width modulation), and more.
Built-in buttons	RESET and BOOT buttons
Built-in LEDs	built-in blue LED connected to GPIO2; built-in red LED that shows the board is being powered
USB to UART bridge	CP2102

This particular ESP32 board comes with 36 pins, 18 on each side. The number of available GPIOs depends on your board model. To learn more about the ESP32 GPIOs, read our GPIO reference guide:

<https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>



1. It comes with a microUSB interface that you can use to connect the board to your computer to upload code or apply power.
2. It uses the CP2102 chip (USB to UART) to communicate with your computer via a COM port using a serial interface. Another popular chip is the CH340. Check what's the USB to UART chip converter on your board because you'll need to install the required drivers so that your computer can communicate with the board (more information about this later in this guide).
3. This board also comes with a RESET button (may be labeled EN) to restart the board and a BOOT button to put the board in flashing mode (available to receive code). Note that some boards may not have a BOOT button.
4. It also comes with a built-in blue LED that is internally connected to GPIO 2. This LED is useful for debugging to give some sort of visual physical output. There's also a red LED that lights up when you provide power to the board.



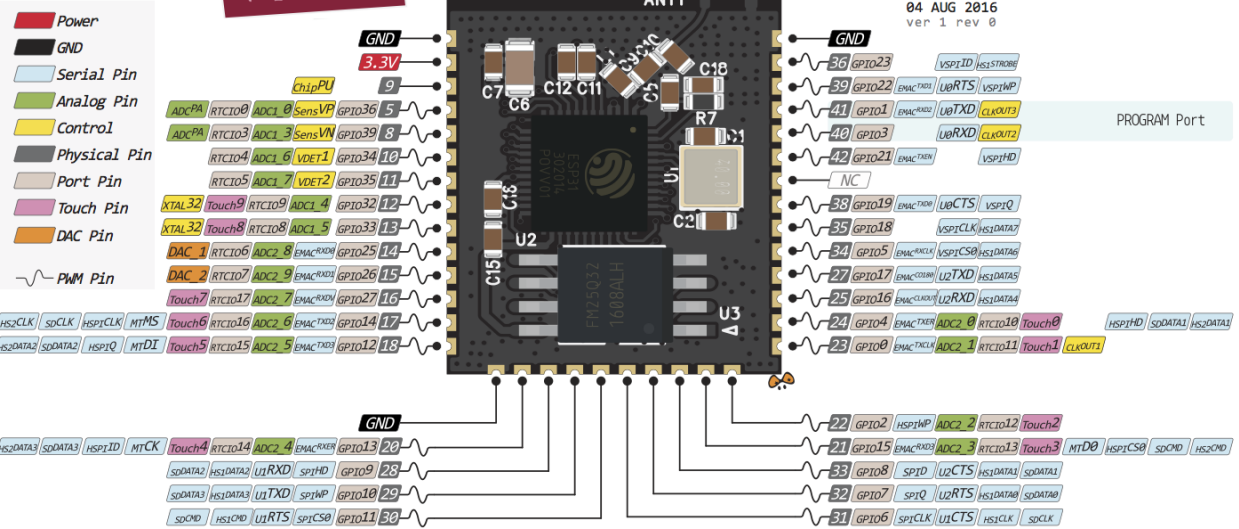
## 2.7 ESP32 GPIOs Pinout Guide

The ESP32 chip comes with 48 pins with multiple functions. Not all pins are exposed in all ESP32 development boards, and some pins should not be used. The ESP32 DEVKIT V1 DOIT board usually comes with 36 exposed GPIOs that you can use to connect peripherals.

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:

# WROOM32

## PINOUT



## **Power Pins**

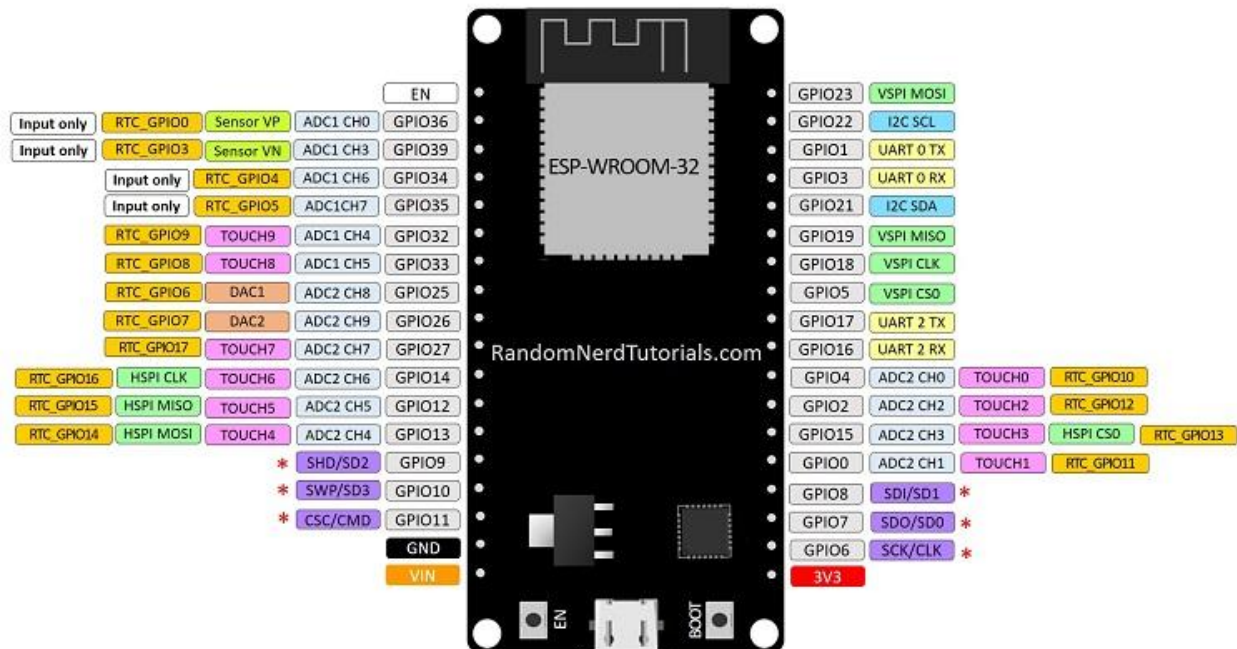
Usually, all boards come with power pins: 3V3, GND, and VIN. You can use these pins to power the board (if you're not providing power through the USB port), or to get power for other peripherals (if you're powering the board using the USB port).

## **General Purpose Input Output Pins (GPIOs)**

Almost all GPIOs have a number assigned and that's how you should refer to them—by their number. With the ESP32 you can decide which pins are UART, I2C, or SPI – you just need to set that on the code. This is possible due to the ESP32 chip's multiplexing feature that allows it to assign multiple functions to the same pin. If you don't set them on the code, the pins will be configured by default as shown in the figure below (the pin location can change depending on the manufacturer). Additionally, there are pins with specific features that make them suitable or not for a particular project.

# ESP32 DEVKIT V1 – DOIT

version with 36 GPIOs



\* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on ESP-WROOM-32 and are not recommended for other uses.

We have a detailed guide dedicated to the ESP32 GPIOs that we recommend you read: [ESP32 Pinout Reference Guide](#). It shows how to use the ESP32 GPIOs and explains what are the best GPIOs to use depending on your project.



The placement of the GPIOs might be different depending on your board model. However, usually, each specific GPIO works in the same way regardless of the development board you're using (with some exceptions). For example, regardless of the board, usually GPIO5 is always the VSPI CS0 pin, GPIO 23 always corresponds to VSPI MOSI for SPI communication, etc.

## 2.8 How to Program the ESP32?

The ESP32 can be programmed using different firmware and programming languages. You can use:

- [Arduino C/C++ using the Arduino core for the ESP32](#)
- Espressif IDF (IoT Development Framework)
- [Micropython](#)
- JavaScript
- LUA
- ...

Our preferred method to program the ESP32 is with C/C++ “Arduino programming language”. We also have some guides and tutorials using [MicroPython firmware](#).

Throughout this guide, we’ll cover [programming the ESP32 using the Arduino core for the ESP32 board](#). If you prefer using MicroPython, please refer to this guide: [Getting Started with MicroPython on ESP32](#).

## **Programming ESP32 with Arduino IDE**



To program your boards, you need an IDE to write your code. For beginners, we recommend using Arduino IDE. While it’s not the best IDE, it works well and is simple and intuitive to use for beginners. After getting familiar with Arduino IDE and you start creating more complex projects, you may find it useful to use [VS Code with the Platformio extension](#) instead.

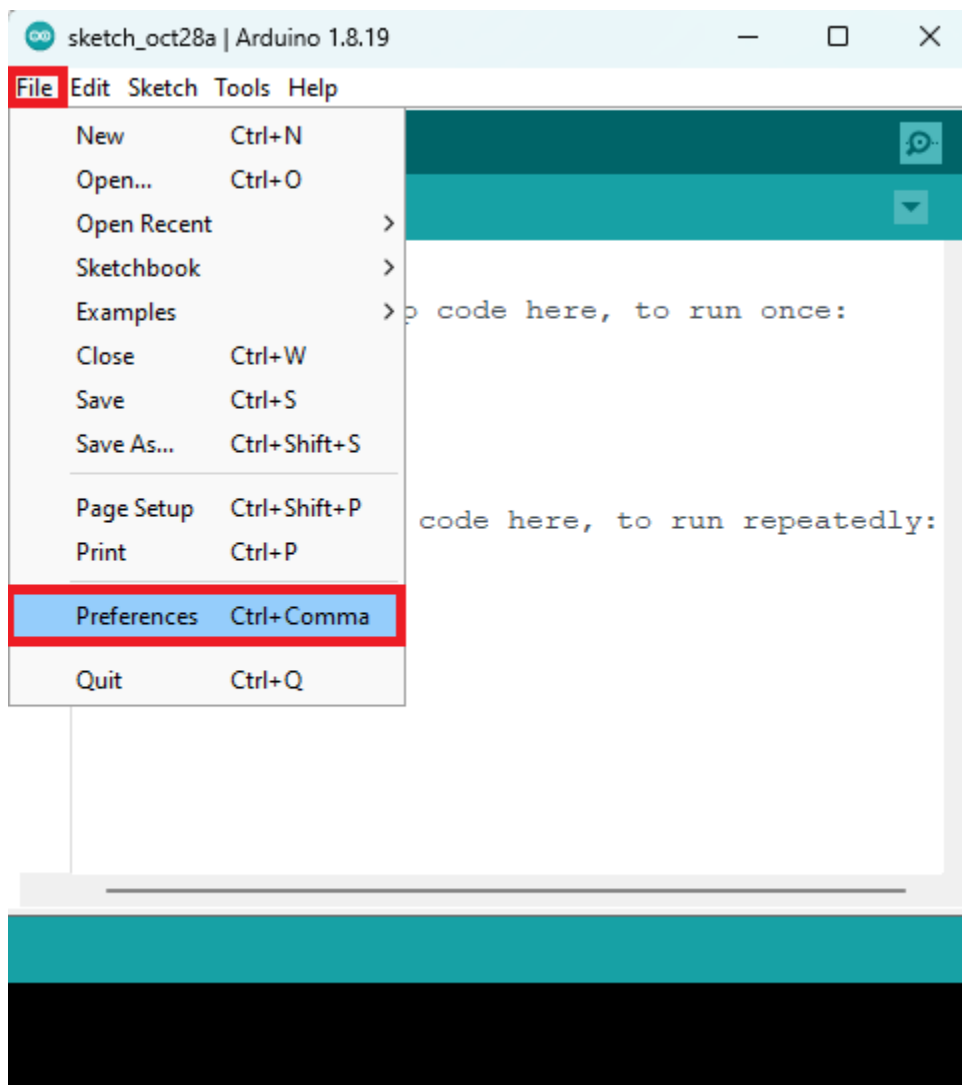


If you're just getting started with the ESP32, start with [Arduino IDE](#). At the time of writing this tutorial, we recommend using the legacy version (1.8.19) with the ESP32. While version 2 works well with Arduino, there are still some bugs and some features that are not supported yet for the ESP32.

## 2.9 Installing the ESP32 in Arduino IDE

To be able to program the ESP32 using Arduino IDE, you need to add support for the ESP32 boards. Follow the next steps:

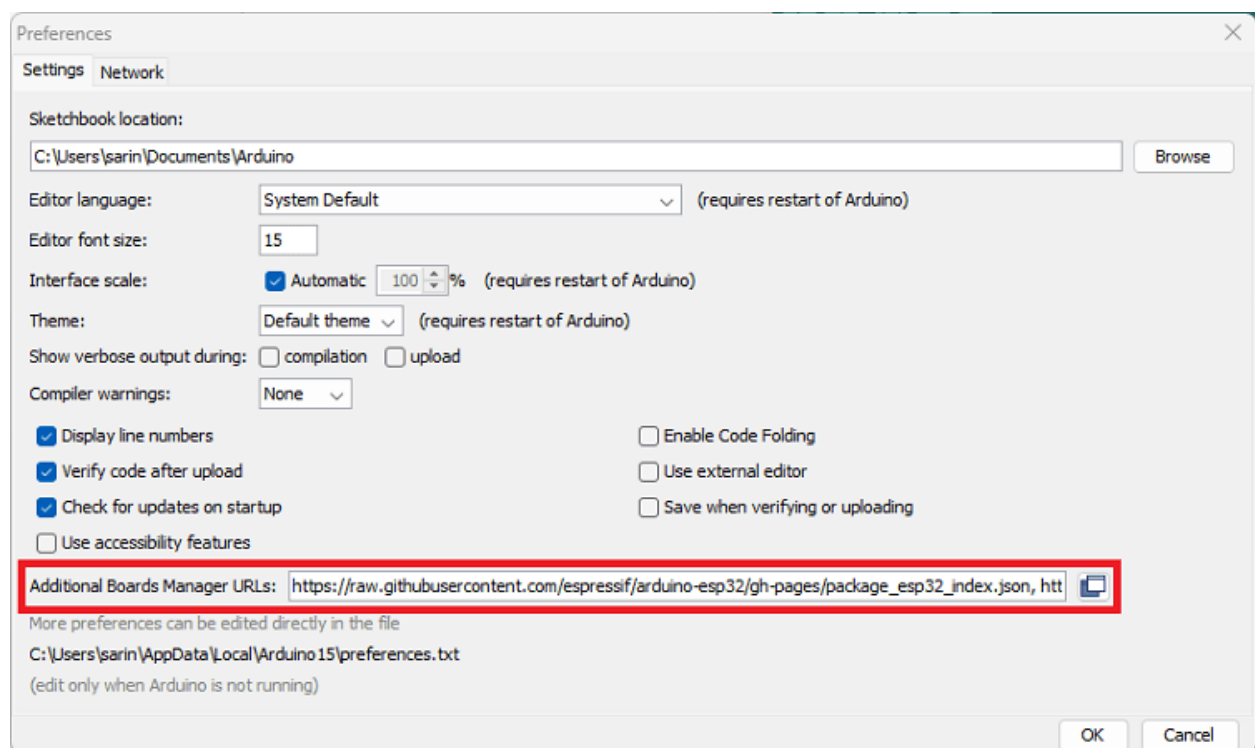
**Go to File > Preferences.**



2. Enter the following into the “*Additional Board Manager URLs*” field.  
This will add support for ESP32 and ESP8266 boards as well.

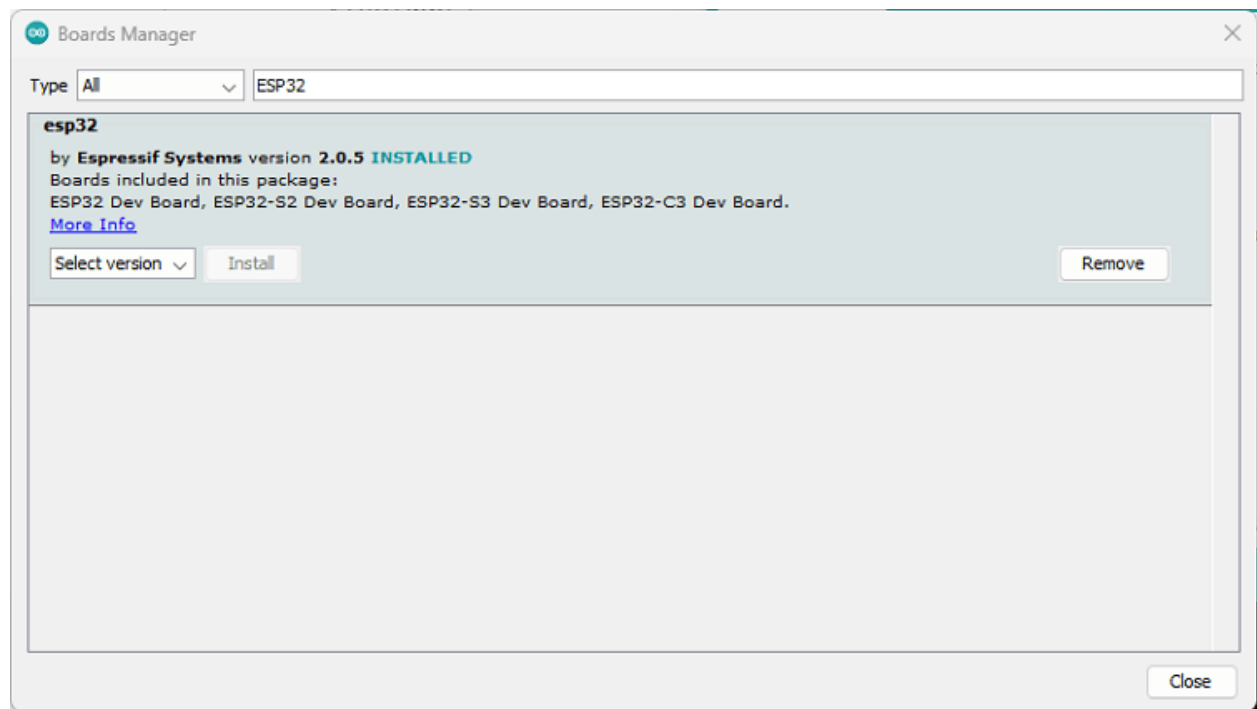
```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json,  
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

See the figure below. Then, click the “OK” button.



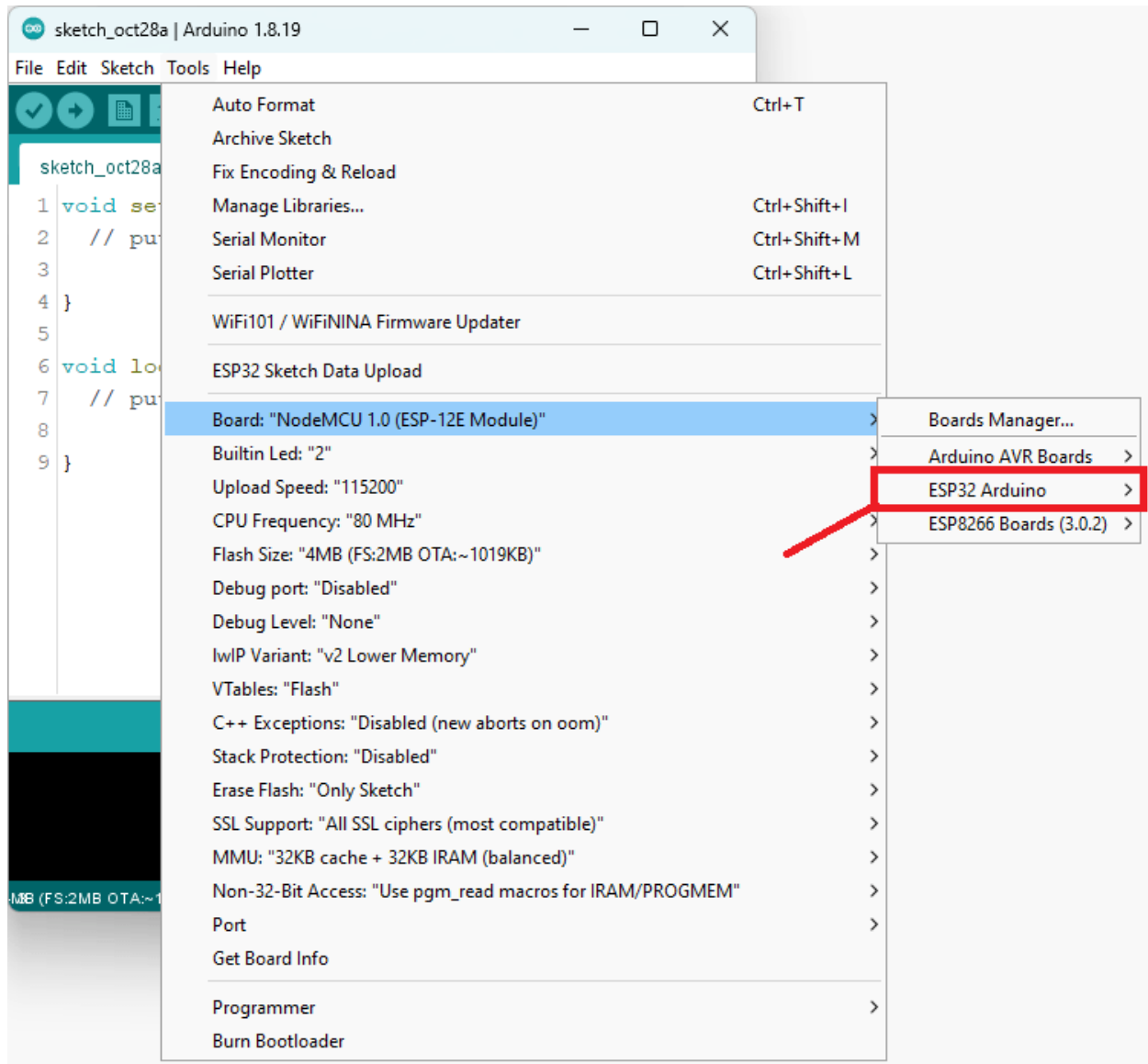
3. Open the Boards Manager. Go to Tools > Board > Boards Manager...
4. Search for ESP32 and install the “ESP32 by Espressif Systems”:

That’s it. It will be installed after a few seconds.



After this, restart your Arduino IDE.

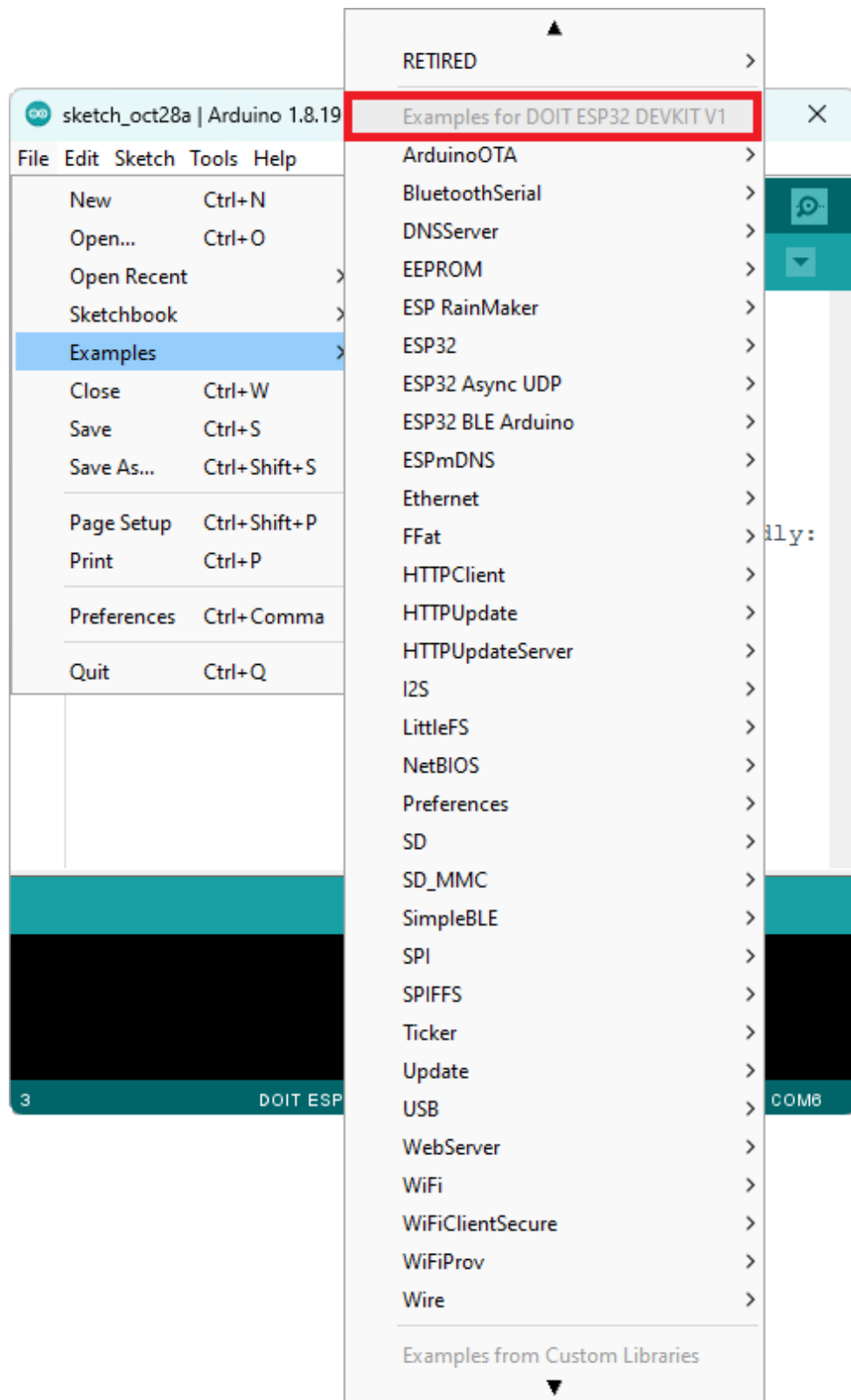
Then, go to Tools > Board and check that you have ESP32 boards available.



Now, you're ready to start programming your ESP32 using Arduino IDE.

## 2.10 ESP32 Examples

In your Arduino IDE, you can find multiple examples for the ESP32. First, make sure you have an ESP32 board selected in Tools > Board. Then, simply go to File > Examples and check out the examples under the ESP32 section.



## **Update the ESP32 Core in Arduino IDE**

Once in a while, it's a good idea to check if you have the latest version of the ESP32 boards add-on installed. You just need to go to Tools > Board > Boards Manager, search for ESP32, and check the version that you have installed. If there is a more recent version available, select that version to install it.

### **2.11      Upload Code to the ESP32 using Arduino IDE**

- To show you how to upload code to your ESP32 board, we'll try a simple example available in the Arduino IDE examples for the ESP32.
- First, make sure you have an ESP32 selected in Tools > Board. Then, go to File > Examples > WiFi > WiFiScan.
- This will load a sketch that scans Wi-Fi networks within the range of your ESP32 board.



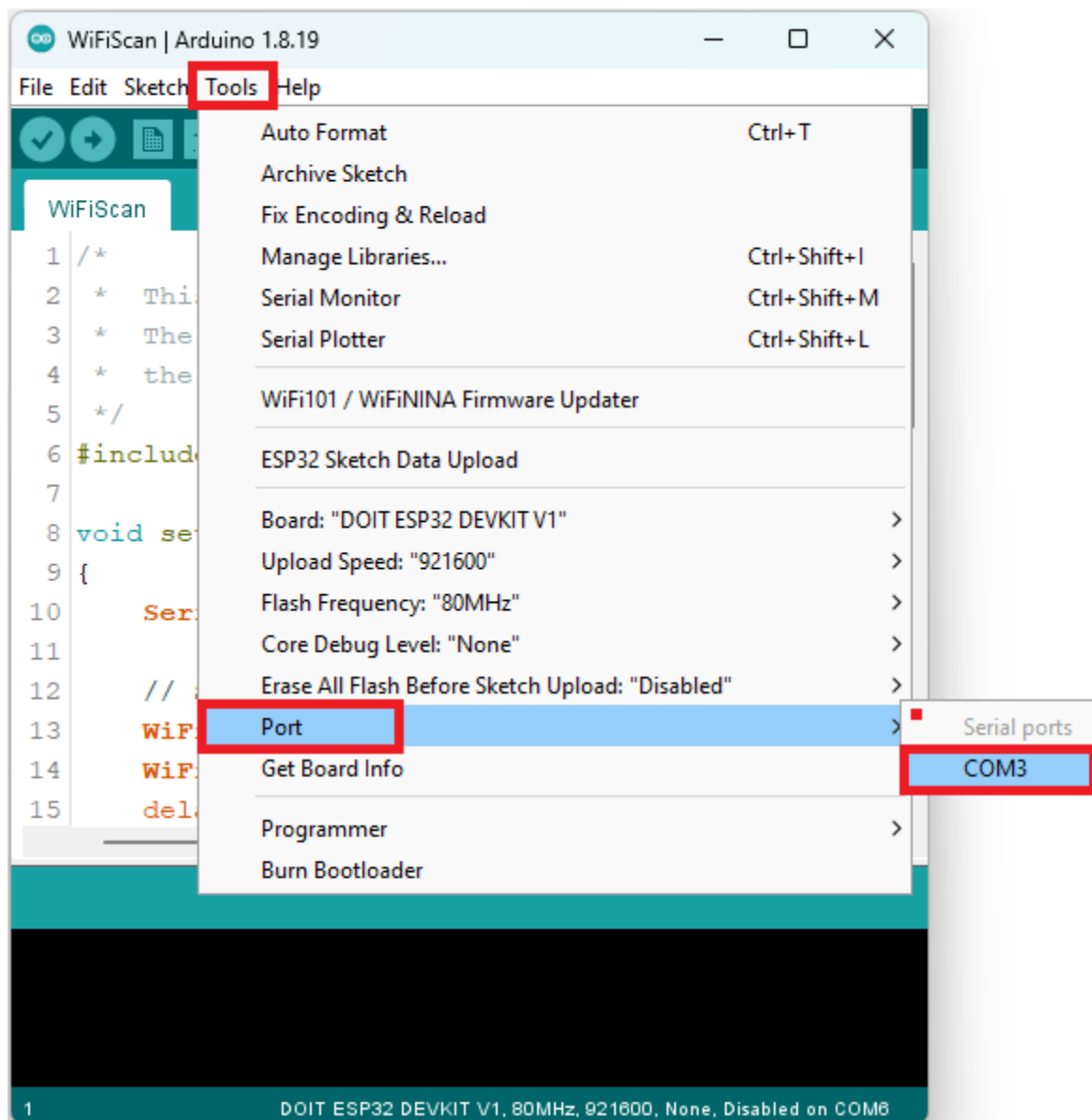
Connect your ESP32 development board to your computer using a USB cable. If you have an ESP32 DEVKIT DOIT board, the built-in red LED will turn on. This indicates the board is receiving power.

**Important:** You must use a USB cable with data wires. Some USB cables from chargers or power banks are power only and they don't transfer data—these won't work.

**Now, follow the next steps to upload the code.**

1) Go to Tools > Board, scroll down to the ESP32 section and select the name of your ESP32 board. In my case, it's the DOIT ESP32 DEVKIT V1 board.

2) Go to Tools > Port and select a COM port available. If the COM port is grayed out, this means you don't have the required USB drivers. Check the section [Installing USB Drivers](#) before proceeding.



**3) Press the upload button.**





Some boards will automatically go into flashing mode and the code will be successfully uploaded straight away.

Other boards don't go into flashing mode automatically, so you may end up getting the following error.

Failed to connect to ESP32: Timed out... Connecting...

Or something like:

A fatal error occurred: Failed to connect to ESP32: Wrong boot mode detected (0x13)! The chip needs to be in download mode.

This means the ESP32 was not in flashing mode when you tried to upload the code. In this situation, you should long press the board BOOT button, when you start seeing the "Connecting...." message on the debugging window.

Note: in some boards, a simple trick can make the ESP32 go into flashing mode automatically. Check it out on the following tutorial: [\[SOLVED\] Failed to connect to ESP32: Timed out waiting for packet header](#).

Now, the code should be successfully uploaded to the board. You should get a "Done uploading "message".

```
Done uploading.
```

```
Wrote 681936 bytes (444520 compressed) at 0x00010000 in 7.0 s  
Hash of data verified.
```

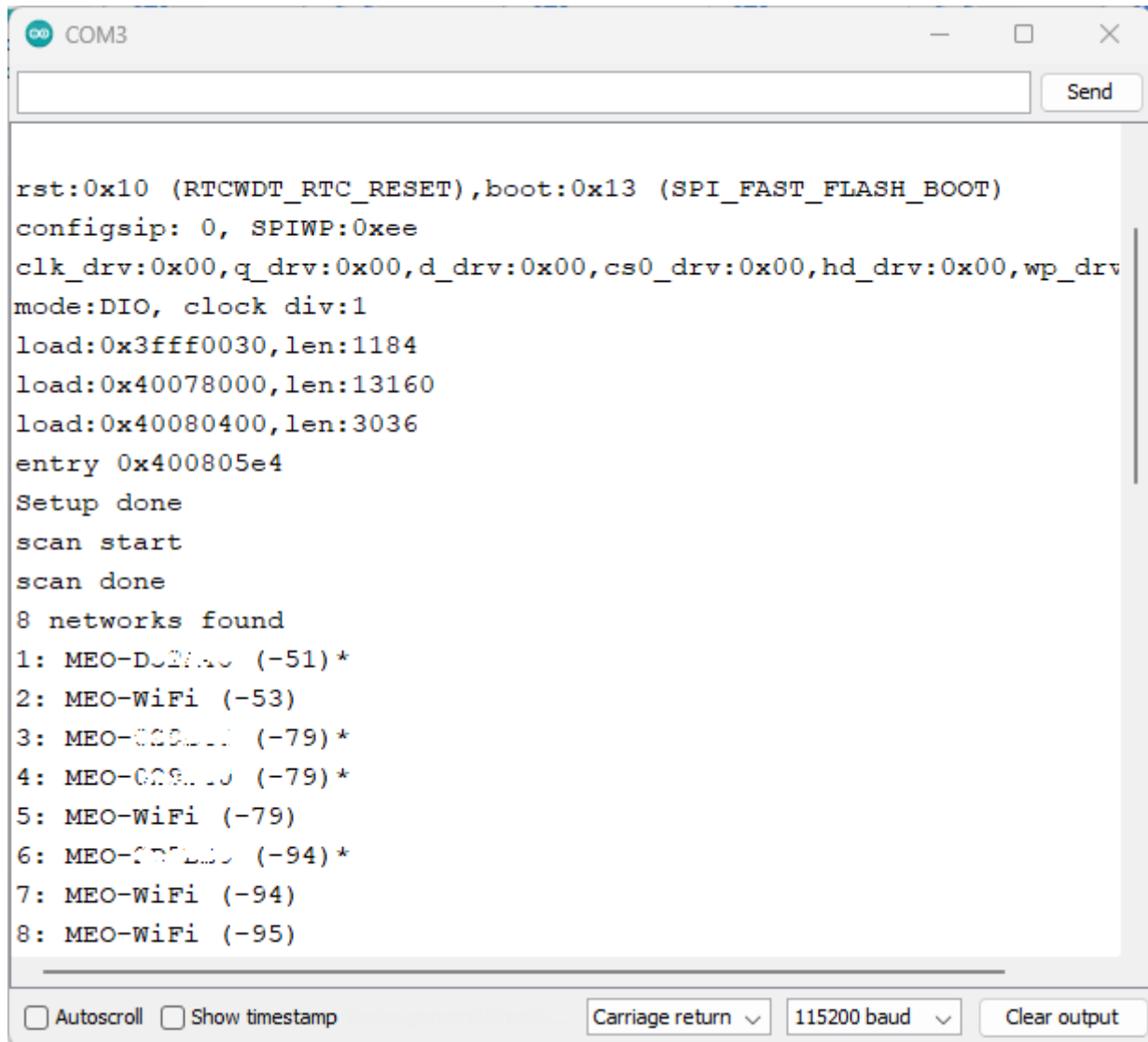
```
Leaving...
```

```
Hard resetting via RTS pin...
```

## 2.12 Demonstration

To see if the code is working as expected, open the Serial Monitor at a baud rate of 115200. Press the ESP32 RST or EN button to restart the board and start running the newly uploaded code.

You should get a list of nearby wi-fi networks.



```
COM3

rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv
mode:DIO, clock div:1
load:0x3fff0030,len:1184
load:0x40078000,len:13160
load:0x40080400,len:3036
entry 0x400805e4
Setup done
scan start
scan done
8 networks found
1: MEO-D022110 (-51)*
2: MEO-WiFi (-53)
3: MEO-0000110 (-79)*
4: MEO-0000110 (-79)*
5: MEO-WiFi (-79)
6: MEO-0000110 (-94)*
7: MEO-WiFi (-94)
8: MEO-WiFi (-95)

☐ Autoscroll ☐ Show timestamp
Carriage return ▼ 115200 baud ▼ Clear output
```

This means everything went as expected.

## Installing ESP32 USB Drivers

After connecting the ESP32 board to your computer, if the COM port in Arduino IDE is grayed out, it means you don't have the necessary USB drivers installed on your computer.

Most ESP32 boards either use the CP2101 or CH340 drivers. Check the USB to UART converter on your board, and install the corresponding drivers.

You'll easily find instructions with a quick google search. For example "install CP2101 drivers Windows".

## Wrapping Up

We hope you've found this getting started guide useful. I think we've included all the required information for you to get started. You learned what is an ESP32, how to choose an ESP32 development board, and how to upload new code to the ESP32 using Arduino IDE.



Want to learn more? We recommend the following tutorials to get started:

- [ESP32 Digital Inputs and Digital Outputs \(Arduino IDE\)](#)
- [ESP32 Web Server Tutorial](#)

Also, don't forget to take a look at the ESP32 pinout to learn how to use its GPIOs:

- [ESP32 Pinout Reference: Which GPIO pins should you use?](#)

If you're serious about learning about the ESP32, we recommend taking a look at our best-selling eBook:

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

You can also check all our free ESP32 tutorials and guides on the following link:

- [More ESP32 Projects](#)

# **Chapter – 3**

## **Internet Of Things (IoT)**

# Python

## What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

### It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

## What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

## Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

## Python Install

Many PCs and Macs will have python already installed. To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have Python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

## # Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

## Key Features

### Indentation

Python uses indentation to indicate a block of code. Python will give you an error if you skip the indentation:

You have to use the same number of spaces in the same block of code, otherwise Python will give you an error:

Comments starts with a #, and Python will ignore them:

### Creating Variables

Python has no command for declaring a variable. A variable is created the moment you first assign a value to it.

### Casting Variable

If you want to specify the data type of a variable, this can be done with casting.

example

```
x = str(3)  # x will be '3'  
y = int(3)  # y will be 3
```



```
z = float(3) # z will be 3.0
```

- You can get the data type of a variable with the `type()` function.
- Variable names are case-sensitive.

## Multi Words Variable Names

Variable names with more than one word can be difficult to read. There are several techniques you can use to make them more readable:

### Camel Case

Each word, except the first, starts with a capital letter:

```
myVariableName = "John"
```

### Pascal Case

Each word starts with a capital letter:

```
MyVariableName = "John"
```

### Snake Case

Each word is separated by an underscore character:

```
my_variable_name = "John"
```

## Many Values to Multiple Variables

Python allows you to assign values to multiple variables in one line

example

```
x, y, z = "Orange", "Banana", "Cherry"
```

## One Value to Multiple Variables

You can also assign the same value to multiple variables in one line:

example

```
x = y = z = "Orange"
```

## Unpack a Collection

If you have a collection of values in a list, tuple etc. Python allows you to extract the values into variables. This is called unpacking.

```
fruits = ["apple", "banana", "cherry"]
```

```
x, y, z = fruits
```

**#### In the print() function, you output multiple variables, separated by a (,) or '+' operator. But in the print() function, when you try to combine a **\*\*string\*\*** and a *number* with the '+' operator, Python will give you an error:**

**So the best way to output multiple variables in the print() function is to separate them with *commas*, which even support different data types:**



