The Arduino IDE, works great for small applications. However, for advanced projects with more than 200 lines of code, multiple files, and other advanced features like auto completion and error checking, VS Code with the PlatformIO IDE extension is the best alternative.

You can press Ctrl+Shift+P or go to view > Command Palette… to show all the available commands. If you’re searching for a command and you don’t know where it is or its shortcut, you just need to go to the Command Palette and search for it.

**PlatformIO IDE Overview.**

Uploading a “Blinking LED” sketch to ESP32.

VS Code and PlatformIO have a folder structure that is different from the standard .ino project.

**Platformio.ini file.**

The platformio.ini file is the PlatformIo Configuration File for your project. It shows the platform, board, and framework for your project. You can also add other configurations like libraries to be included, upload options, changing the Serial Monitor baud rate and other configurations.

* Platform: which corresponds to the SoC used by the board.
* Board: the development board you’re using.
* Framework: the software environment that will run the project code.

With the ESP32 and ESP8266, if you want to use a baud rate of 115200 in your Serial Monitor, you just need to add the following line to your platform.ini file

//

monitor\_speed = 115200

In this file, you can also include the identifier of libraries you’ll use in your project using the lib\_deps directive, as we’’ll see later.

**SRC folder.**

The src folder is your working folder. Under the src folder, there’s a main.cpp file. That’s where you write your code.

In PlatformIO, all your Arduino sketches should start with the #include <Arduino.h>.

Uploading Code using PlatformIO IDE:

ESP32.

Click on the upload icon to compile and upload the code.

**Detect COM Port.**

PlatformIO will automatically detect the port your board is connected to. To check the connected devices you can go to the PIO Home and click the Devices icon.

**Troubleshooting.**

If when trying to upload code you get the following error: “Failed to connect to ESP32: Timed out waiting for packet header” it usually means that your board is not in flashing mode when you’re uploading the code.

When this happens you need to press the ESP32 on-board BOOT button when you start seeing a lot of dots in the debugging window.

If you don’t want to have to press the BOOT button every time you upload new code, you can follow this guide. <https://randomnerdtutorials.com/solved-failed-to-connect-to-esp32-timed-out-waiting-for-packet-header/>

**Changing the Serial Monitor Baud rate- PlatformIO IDE.**

The default baud rate used by PlatformIO is 9600. However, it is possible to set up a different value as mentioned previously. On the File Explorer, under your project folder, open the platform.ini file and add the following line:

//

monitor\_speed = baud\_rate

**Installing ESP32 Libraries on PlatformIO IDE.**

Click the Home ison to go to PlatformIO Home. Click on the Libraries icon on the left side bar.

Search for the library you want to install.

Click on the library you want to include in your project. Then, click Add to Project.

Select the project were you want to use the library.

This will add the library identifier using the lib\_deps directive on the platform.ini file. If you open your project’s platform.ini file you will see..

Alternatively, on the library windows, if you select the installation tan and scroll a bit, you’ll see the identifier for the library. The library identifiers are highlighted in red.

If you need multiple libraries, you can separate their name by a coma or put them on different lines.

PlatformIO has a built-in powerful Library Manager, that allows you to specify custom dependencies per project in the Project Configuration File platform.ini using the lib\_deps. This will tell PlatformIO to automatically download the library and all dependencies when you save the configuration file or when you compile your project.

Here’s some of the advantages of using VS Code with PlatformIO IDE over Arduino IDE:

* It detects the COM port your board is connected to automatically;
* VS Code IntelliSense: Auto-Complete. IntelliSense code completion tries to guess what you want to write, displaying the different possibilities and provides insight into the parameters that a function may expect;
* Error Highlights: VS Code + PIO underlines errors in your code before compiling;
* Multiple open tabs: you can have several code tabs open at once;
* You can hide certain parts of the code;
* Advanced code navigation;