

Forum:[Hack Forums](#)

Thread: [\[#1 Tutorial Series\] Introduction to Professional Arduino Development with PlatformI](#)

HerrWinston - July 27, 2026, 2:50 am

PlatformIO - How to build Arduino Projects on a professional level

This is the first thread of my Tutorial Series, stay tuned for further tutorials going much further into specific actuators, sensors, simulations and Codes.

I recommend to first read: <https://hackforums.net/showthread.php?tid=3747830> , it wont be as important for our current Thread topic, but its still an awesome thread to look into!

1. What`s PlatformIO?

PlatformIO is an open-source ecosystem, designed to simplify and improve the development of microcontroller based projects (the ones we will tackle in the future!). While many beginners start with the Arduino IDE, it feels old and misses a lot of features, for example: PlatformIO is more modern, scalable and has many features you just cant miss out on.

Its an extension for Visual Studio Code and combines project management, compilation, library management, uploading and serial communication, if we compare this to the Arduino IDE, its just the same with more features, a cleaner interfac and the benefits of VSC. The project organization of PlatformIO makes larger projects easier to maintain, as you can divide your code into multiple .cpp files without issues and put them together in the main file.

2. Why change from Arduino IDE to platformIO

I have basically given out the reasons in the introduction of PlatformIO but lets get a little into detail: Arduino IDE is good for simpler projects and I used it a lot before I learned about platformIO aswell but the bigger the projects get the more you wish for something more convinient.

- PlatformIO has the benefits of VSC, that Arduino IDE misses, for example: syntax highlighting, auto completion and navigation features (for variables this was a game changer for me)
- Better Project organization: simply splitting my project into multiple .cpp and .h files instead of having a single .ino is something I learned to love, much easier to find problems through this and the code is easily understandable even after not looking at it for a month
- Automatic Library Management, installing and managing libraries directly in the project is convinient but why this is such a good feature will be shown in the correct section of this thread.
- Managing multiple Projects is fairly simple and doesnt need need you to browse through your folders for an hour.

3. How to integrate it into VSCode / how to install

This section will cover the full installation of PlatformIO and hopefully you guys find a good youtube video how to install VSC, because I will assume everyone here has it atleast installed.

Step 0: Install VSC and launch it.

Step 1: Open the Extensions Marketplace

Click the Extensions Icon on the left sidebar, or be a power user and press Ctrl + Shift + X.

In the search bar that pops up type in: PlatformIO IDE

It should be the first one to pop up, install it and wait a few minutes, as it automatically installs all required components aswell.

Step 2: Restart VSC completely, then restart it.

You should now see the PlatformIO icon on the same bar we just used to get into the extension marketplace.

? loading encrypted image...

Click on this to get into the dashboard, congrats you just installed PlatformIO. Now its time to understand it!

4. Explaining PlatformIO dashboard

Now that we have opened the dashboard, alot of things jump into our sight.

We will ignore all of it and click on "Open"

This will open the actual Dashboard.

? loading encrypted image...

We have a few things we see right off the start:

Recent News: shows us newly published libraries, articles explaining embedded coding or the newest projects made by the community

Right below is Recent Projects: this place will be most likely empty for you, FOR NOW!

Quick Access: this will become important very soon. We can make completely new Projects (foreshadowing), Import projects from the Arduino IDE if your converting from it to PlatformIO. Open a PlatformIO project or check out the project examples that you may also find in the Arduino IDE, with a few more going more into detail for the ESP32.

On the left we have another bar, home is where we are currently at.

Projects: this is the big folder where all our projects will be stored and configurable.

Inspect: lets us check a project, the code, flash use, ram use, defects, it explains errors and gives detailed explanation if speed or project size start to truly matter.

Libraries: this is where we will get all our libraries for bigger projects.

Boards: shows all the supported boards and some useful information right next to them

Platforms: decide what platforms you need for your microcontroller

Devices: when connecting a device to your pc, it gets detected, you get info about it and know everything works well on that end.

5. Let`s make your first Project!

Now that we know our way around the Dashboard, its time to understand what my hype has been all about.

We stay at the "home" menu and go to quick access, there we will click the Button "New Project"

This will open:

? loading encrypted image...

1. Name: Give your Proct a name, for this thread / test I would recommend "Blinking LED"

2. Board: Choose the Board you have, dont worry too much, if you use an Arduino Uno R3 you still choose the Arduino Uno, for example if you have an ESP32 live mini kit, you just choose esp32 devkit and take whatever, it truly doesnt matter THAT much, personally I recommend: uPesy ESP32 Wroom DevKit.

3. Framework: This is important, but if you code with Arduino or Esp32 you can still stay at the

Arduino Framework.

4. Finish : This will take a few seconds and you successfully created your first project!

6. Explaining the Project structure

Wow, we instantly got thrown into a Project structure with 6 Folders and a ton of files... HELP!

Alright, so lets first read through them and relax.

I admit, this looks so much more complicated than Arduino IDE, but this will help maintain your Projects and everything has a specific purpose we will need.

? loading encrypted image... This may look confusing at first, but you really only need 2, especially at the beginning.

Lets go through them quickly!

.pio = Auto generated file from PlatformIO, you normally never need to look into this, it contains cached libraries, compiled files and so on.

.vscode = Stores vsc configurations for this project, normally you dont open this either.

include = this is where we will put our own header (.h) files these files hold class definitions, constants, and shared variables that are used across multiple source files. I personally use this in every project, it will help keep your projects modular

lib = This folder is intended for your own custom libraries, personally I use these rarely, but you can input reusable modules to keep your projects more efficient.

src = Most important Folder of them all, this is where the main.cpp is located and where your application code lives. By default only main.cpp is in here but as your project grows you may use multiple .cpp files (this is where the include folder comes in handy)

test = This is for unit testing, a solo dev / hobbyist wont use this, I havent either.

.gitignore = tells git which files not to include into the upload to your repository.

platformio.ini = The config file of your project. How this works will be shown in the next topic.

So in the end, it really only looks like this:

? loading encrypted image...

Just as a reminder: in main.cpp the magic happens, if it comes down to it, you could ignore every other folder and file :)

7. platformio.ini whats that?

This file is the heart of every project. It contains all the configurations and tells PlatformIO how to build and upload your projects!

? loading encrypted image...

Here we can configure a few things:

The framework, the board and platform, if we choose to switch this up, normally this is defined from what we input while creating the project!

Now lets get into what we can add to this file as extra configurations.

The 2 most important additions:

monitor_speed | this changes the Serial monitor Baud Rate, 9600 is the average for Arduino Projects while 115200 is the modern rate, most ESP32`s use!

lib_deps | this is where you can manually input Libraries your project may need.

also implementable if needed: upload_speed , upload_port, build_flags, but most of this is rather unimportant for now

Here is an example of what we could implement:

? loading encrypted image...

This one changes the serial monitor speed to the Arduino Uno average and inputs a library for

configuring Neo Pixels, we wont need it, but its a nice show off :)

if you want to integrate it into your own project:

```
Code [env:uno]
platform = atmelavr
board = uno
framework = arduino
monitor_speed = 9600
lib_deps =
  adafruit/Adafruit NeoPixel
```

8. How do I integrate libraries?

PlatformIO offers multiple options for integrating libraries, one of which we have seen already, inputting it directly into the .ini file, the library would automatically download once saving the file. This is imo recommended if you know the lib already, as its automatically updated too, also through the library manager or custom libraries, in this section I will explain the library manager.

Step 1. We click on the Alien symbole again on the side

Step 2. Click "Open" or Click on "Libraries" and skip the next step

Step 3. click on "Libraries" in your dashboard

Step 4. Search for the library you need, for this example, we will simply search WifiManager by tzapu

Step 5. click on the correct library and "Add to Project"

This will open:

? loading encrypted image...

Step 6. We wont change anything, except "Select a project" here, scroll until you find our project.

Step 7. go into the .ini file and verify its there

Congrats you now know how to integrate your libraries in PlatformIO!

9. Upload Options and Serial Monitor

Lets get going to build our first PlatformIO project!

This will be the section you will realise why, this being a vsc extension is such an advantage!

We go into the src folder and open the main.cpp file, this code should open up:

```
Code #include <Arduino.h>
```

```
// put function declarations here:
```

```
int myFunction(int, int);
```

```
void setup() {
```

```
    // put your setup code here, to run once:
```

```
    int result = myFunction(2, 3);
```

```
}
```

```
void loop() {
```

```
    // put your main code here, to run repeatedly:
```

```
}
```

```
// put function definitions here:
```

```
int myFunction(int x, int y) {
```

```
    return x + y;
```

}We see one odd thing: `#include <Arduino.h>`, Winston why do we have to include our own code? Well, this is a must have in every code, its a little like `iostream` in `c++`, we have to include it to write our code!

Apart from that, lets just delete it all

and now I really hope you checked out the thread I mentioned at the beginning!

for this project, we will just make a basic code to check how good it works, mine is:

Code `#include <Arduino.h>`

```
const int wait = 300;          // the time between the blinking :)
const int led = 4;             // whatever Pin you want to use

void setup()
{
  pinMode(led,OUTPUT);
  Serial.begin(9600);          // we want to showcase the Serial monitor, so this is sadly needed
}

void loop()
{
  digitalWrite(led,HIGH);
  Serial.println("LED: 1");
  delay(wait);
  digitalWrite(led,LOW);
  Serial.println("LED: 0");
  delay(wait);
}
```

}Alright, now lets get into the actual point of this section.

On the Top Right of VSC, will be 2 Buttons, on the left and to the right of my Chatgpt symbol. If we go ahead and click on the left one we have this:

? loading encrypted image...

Build: This compiles your code for the selected microcontroller without uploading it, this helps to find errors or see if it compiles correctly into machine code.

Upload: This first builds the project and then uploads the compiled firmware to your microcontroller.

Test: Runs the tests in `test/` folder, so nothing important for us as of now.

Clean: Deletes all builds in `.pio`, which forces a completely fresh build at the next time, personally never use it tho.

Now the Serial Monitor

Click on the symbole next to the Upload options, or on my screenshot, the one right to the Chatgpt symbol.

For this to work, we first have to Upload our code onto the Devboard.

Once the terminal shows this:

? loading encrypted image...

we can go into the Serial Monitor, which you may already know, congrats you now understand how to use the serial monitor and the different Upload options of PlatformIO.

10. Useful Information and Tips/Tricks

Lets explain a few things you should know before!

Include folder and Header files, why are they so cool?

Alright, so lets imagine we have a project with multiple .cpp files and want to have a modular code to work with. then its time to look into header files.

My average use case is an extra folder in src called testing, in that i normally input other .cpp files I use to save code ideas and tests that i can later on build together

Lets say you made an extra .cpp file in src, for example menu.cpp, how will we integrate our functions into the main.cpp? with header files!

Go into the include folder and make a new file, lets call it, menu.h

? loading encrypted image...

Alright, now what is that code I implemented into the menu.h?

#pragma ONCE - if you reference the functions in multiple cpp files it wouldnt work, this pragma is why it doesnt matter how often you reference it, it will only be used once.

void - this is the function in menu.cpp i wanna include into my main.cpp, you can input here many other functions aswell by just writing void ...(); void ...();

In the main.cpp you will then write:

Code #include "menu.h" this will be very useful and something you should include in bigger projects :)

Another Tip: Just dont touch the .pio folder, its not worth it....

Some Shortcuts:

CTRL + ALT + U = Upload

CTRL + ALT + B = Build

CTRL + ALT + S = Serial monitor

CTRL + SHIFT + P = Open Command Palette - use this for Terminal: Kill all Terminals, if your Serial monitor tells you its already running

CTRL + F = Find - this works wonders if you want to find variables you wanna change.

11. Why I switched to PlatformIO

Let me give you guys the reason I switched:

1. The design - VSCode is fully optimised to look as cool as I want it, while Arduino IDE is too plain and boring

2. Auto completion - Arduino IDE does not have this feature and i desperately needed it as a starter, who remembers that pinMode has only the M capitalised?

3. Syntax Highlighting - in PlatformIO my code looks organized and color coded too, this is very useful and was a neat thing to have. (noticed the IDE has this too now, just looks worse)

4. Library management is just easier, the auto updates and easily being able to change the needed libs in the .ini is awesome

5. Project management - the main reason YOU should migrate, no need to write it all in 1 file if you can write it in 10 files you can easily check to understand your code again.

And in general, just look at this:

? loading encrypted image...

This was my first Tutorial of this Series, in the next one we will speak about simulators like tinkercad and give them a thorough review!

Please leave a review of what should be improved in the upcoming editions!

I will try to post one once a week as they take a lot of time and research to be done ^^

A big shoutout to [CL1B](#) <- the man, the myth the legend

[Wake](#) <- His ESP32 tutorials are perfect, check them out once you understand the basics, you will learn a lot!

[Terry A. Davis](#) <- thanks for being my introduction and supporter in embedded system development :)

There are many more that would deserve to get their names on my threads, but I truly appreciate you guys! and if you want to be tagged in the next threads, just tell me in PM or here on thread!

Questions? Post below.

Pr0m3th3us - July 28, 2026, 5:59 pm

Very useful thread for when i go more in depth with arduinos. So far i only have 1, got it a couple months ago and it was the build your own kit. I use it in conjunction with a hat CYD and thermo to keep track of the temp in my PC room. Currently its at 85.9f.

HerrWinston - July 29, 2026, 2:40 am

(Jul 28, 2026 05:59 PM)Pr0m3th3us Wrote: [\[CITE\]](#) Very useful thread for when i go more in depth with arduinos. So far i only have 1, got it a couple months ago and it was the build your own kit. I use it in conjunction with a hat CYD and thermo to keep track of the temp in my PC room. Currently its at 85.9f.

Very nice project! I have built something similar once, using my esp32 and the DHT11 sensor, I built a relatively simple system that monitors the temp and humidity in my room and sends me an update once an hour through a telegram bot, with a little extra work I also got it, that I have a button in the TG Bot that then instantly reads the current measurements and gives them out, with a recommendation if I should open my windows or not.

Was a fun project to build! :)

Pr0m3th3us - July 29, 2026, 10:06 am

(Jul 29, 2026 02:40 AM)HerrWinston Wrote: [\[CITE\]](#) Very nice project! I have built something similar once, using my esp32 and the DHT11 sensor, I built a relatively simple system that monitors the temp and humidity in my room and sends me an update once an hour through a telegram bot, with a little extra work I also got it, that I have a button in the TG Bot that then instantly reads the current measurements and gives them out, with a recommendation if I should open my windows or not.

Was a fun project to build! :)

I also put the DHT11 on it but it was faulty so i need to order another one. Not a big deal though.

CL1B - July 31, 2026, 6:39 am

Good read and intro, I'm not actively following along as I don't own those boards (yet) but I know for sure your series would help those who are new to this and on HF.

Lovely, looking forward to the continuation and perhaps expansion (videos and outside HF?)

Thank you!

HerrWinston - July 31, 2026, 3:18 pm

(Jul 31, 2026 02:51 PM)n0xi0s Wrote: [\[CITE\]](#) woauhheh dude i needed that tut so badly thanks mate

the man, the myth the legend himself

I am glad it helped out! If you have any questions or ideas of threads I could make relating Arduino you can always make a post or hit me up ^^