PlatformIO IDE Installation Guide

PlatformIO works best when installed in Visual Studio Code or as a set of command line tools. The instructions below cover how to use PlatformIO with VS Code.

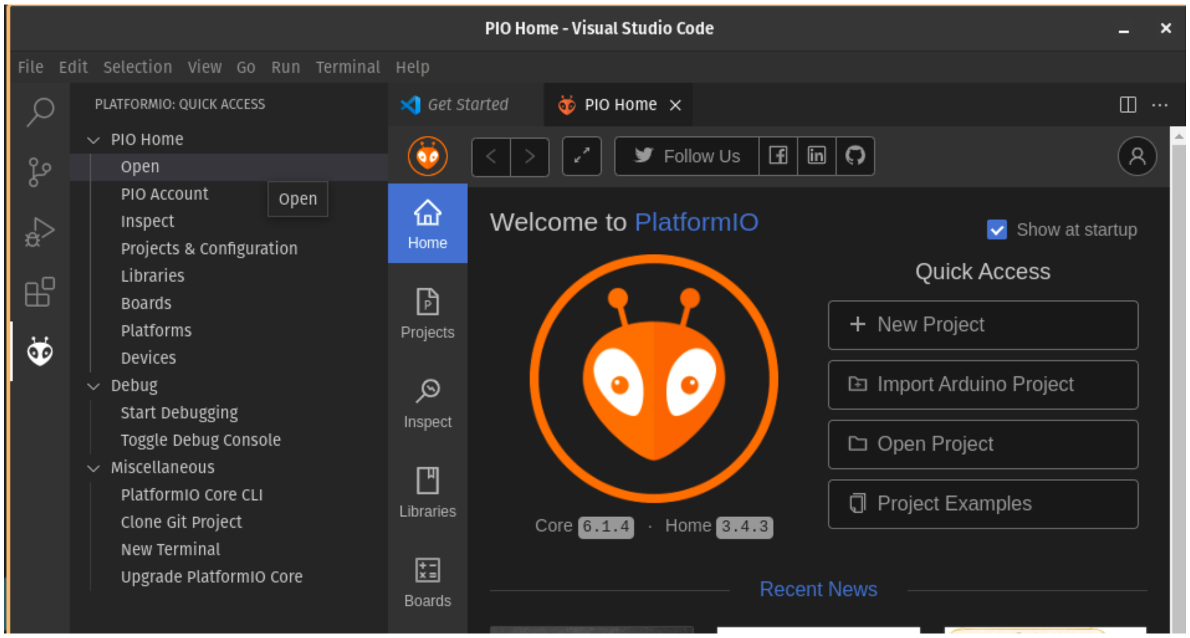
Table of Contents

Install PlatformIO in VS Code

1. Install Visual Studio Code.
2. Follow the steps here to install PlatformIO in VS Code. Alternatively, just search for PlatformIO in extensions and install it.

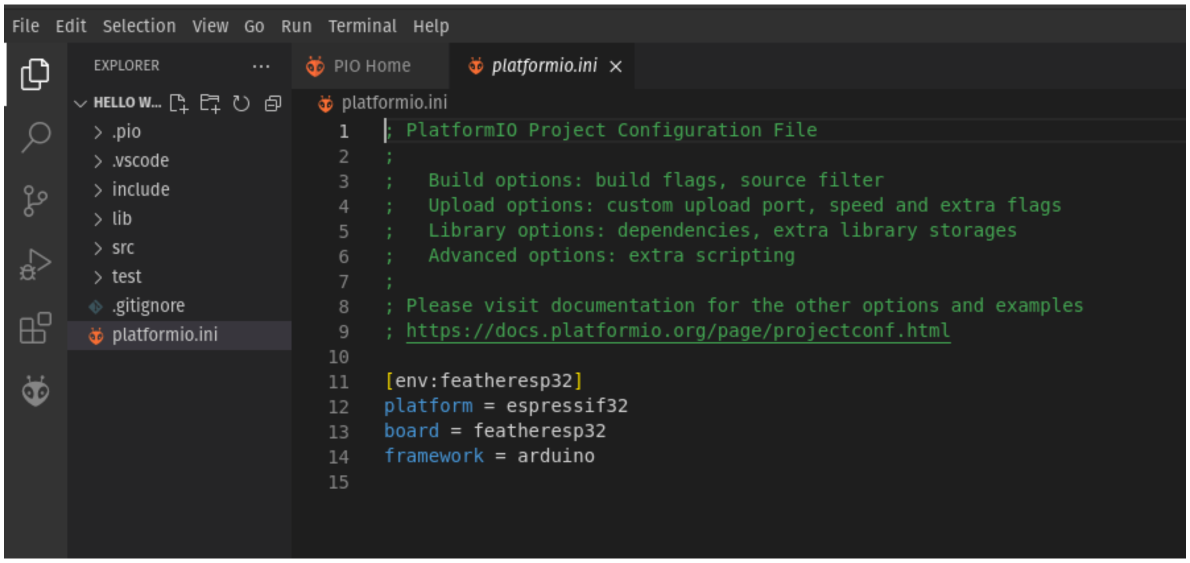
PlatformIO Projects

1. Open VS Code, then click the bug icon on left to open PIO quick access menu.
2. From PIO Home dropdown, select 'Open', this will show the home menu page



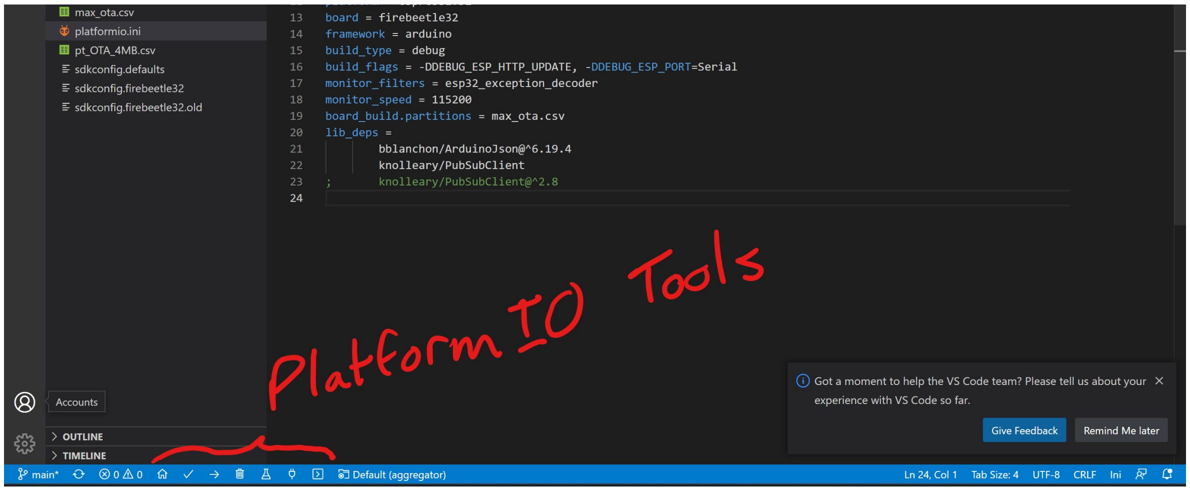
3. Next, click `New Project` from the right hand side of the menu. Use the configuration menu to select the board you are using, which will set the build tools automatically, and also select the Arduino framework. Unselect the toggle at the bottom of the menu to save project in a custom location.
4. After creating the project, it should be opened in VS Code and look something like this:
   - Projects should contain the folders below:

Sub-folder	Contents
`.pio/`	contains output from compilation for various build targets
`include/`	put header files here, contents are automatically added to include search path
`lib/`	put library source code here, meant for larger independent libraries
`src/`	should include the entry point for your program, usually in `main.cpp`
`test/`	should include unit tests for your code base, which can be run using the PlatformIO test module.



```ini
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:featheresp32]
platform = espressif32
board = featheresp32
framework = arduino

```

5. VS Code should recognize this folder as a PlatformIO project since it contains `platformio.ini`. VS Code should reload with the PlatformIO IDE buttons at the bottom of the window. This should happen any time you open a folder with a `platformio.ini` file in it.

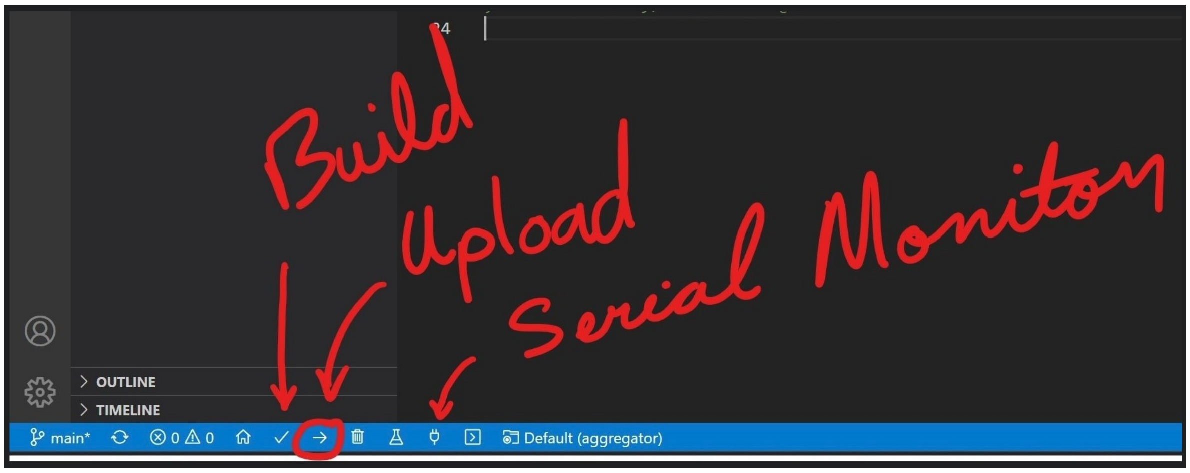


```ini
board = firebeetle32
framework = arduino
build_type = debug
build_flags = -DDEBUG_ESP_HTTP_UPDATE, -DDEBUG_ESP_PORT=Serial
monitor_filters = esp32_exception_decoder
monitor_speed = 115200
board_build.partitions = max_ota.csv
lib_deps =
        bblanchon/ArduinoJson@^6.19.4
        knolleary/PubSubClient
;       knolleary/PubSubClient@^2.8

```

Got a moment to help the VS Code team? Please tell us about your experience with VS Code so far.

6. If you have a micro-controller, connect it to your computer via USB.
7. In VS Code, click the upload button. This should build the code in `src/`, auto-detect the USB port, and upload the executable to the board. You should see terminal output as this process is happening.

- The build button denoted below will compile your code, but not upload
- Upload will build, then upload to the board
- Serial Monitor will open using the configuration options from `platformio.ini` and then show serial data received from USB port.



8. Once the upload has successfully completed, open serial monitor to see serial output from the board (probably looks like a two pronged plug).
    - If errors about accessing the USB device or serial monitor, make sure the Arduino IDE is closed. If errors persist, restart VS Code with admin privileges.
    - If output is unreadable, make sure the baud rate on the serial monitor is set to 115200/9600, whichever your board is running at.

PlatformIO Project Configuration

PlatformIO provides cross platform build tools for thousands of micro-controllers. The build toolchains are implemented or called by automated python scripts installed by PlatformIO depending on the contents of a project's `platformio.ini` file, which marks a directory as a PlatformIO project.

`platformio.ini` is used to specify board (target micro-controller), platform (build tool supplier), framework (standard API to compile for, ex: Arduino), library dependencies, serial monitor options, etc. Additionally, multiple targets (boards) can be specified in the `.ini` file and selected when the project is "built"/compiled. The example below shows some common options that are especially helpful for the ESP32 family of boards, but the link above shows the complete documentation.

Compiled libraries and executables (`.bin`, `.elf`) are stored in the `.pio` directory in the root of the PlatformIO project directory.

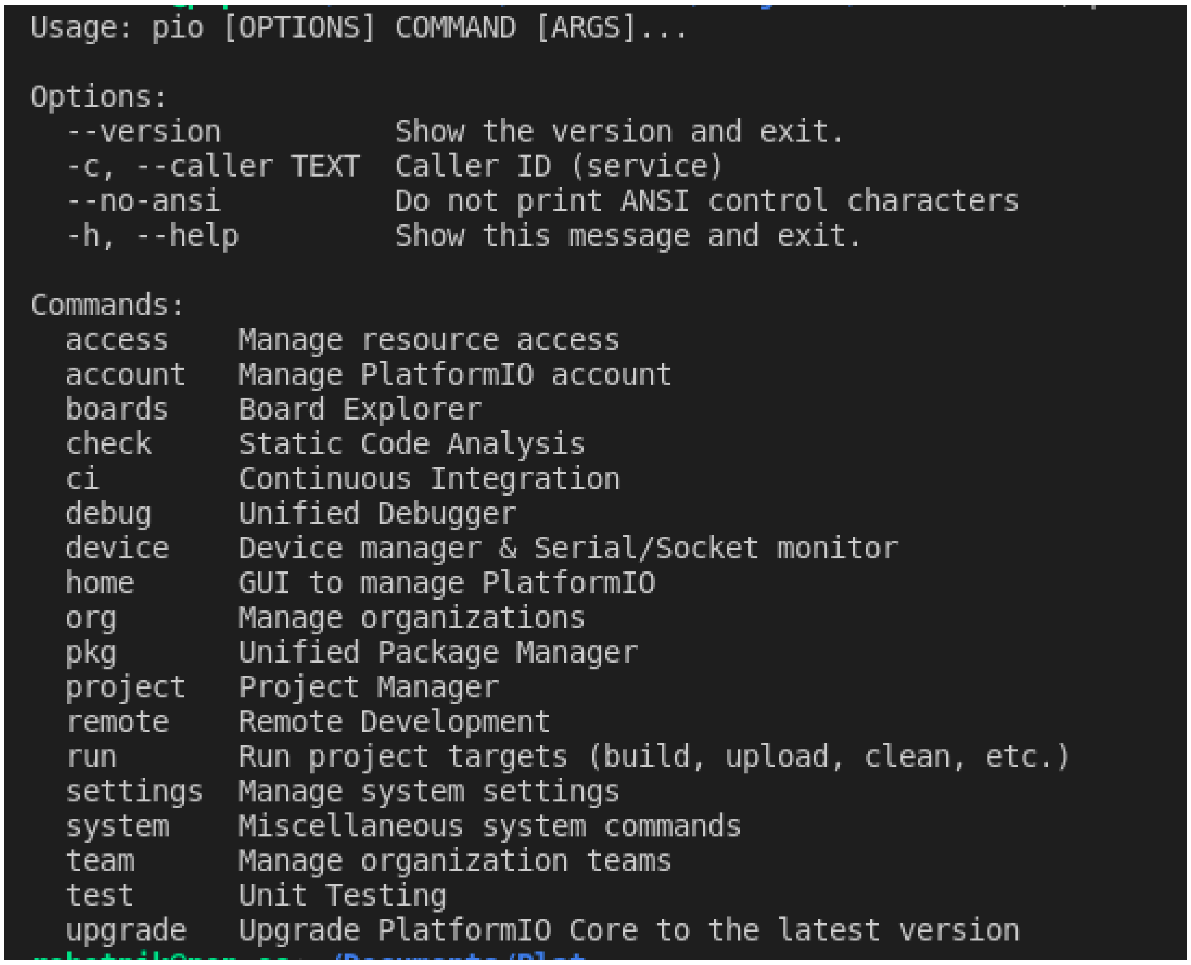
```
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:firebeetle32]
; specify build tools
```

```
platform = espressif32@^5.1.1
; specify board, contains pin definitions, memory sizes
board = esp-wrover-kit
; include the arduino API
framework = arduino
build_type = debug
build_flags = -DDEBUG_ESP_HTTP_UPDATE, -DDEBUG_ESP_PORT=Serial
; esp32_exception_decoder will filter serial monitor output and print stack traces, VERY HELPFUL
monitor_filters = esp32_exception_decoder
; any time serial monitor is opened for this project, default to 115200 baud
monitor_speed = 115200
; can specify custom partition map for board memory, dont do this for most projects
board_build.partitions = max_ota.csv
; example library dependencies, will be installed with specified version
lib_deps =
        bblanchon/ArduinoJson@^6.19.4
        knolleary/PubSubClient
;       knolleary/PubSubClient@^2.8
```

PlatformIO CLI

PlatformIO also includes a command line interface which can be accessed through the VS Code terminal. This can be preferable to the VS Code interface for several reasons:

1. Your project may be built for multiple target boards. VS Code's build/upload options will only build the default target, thus, if you want to quickly switch between build targets, look into using `pio run -t upload -t monitor`, which uploads and opens serial monitor automatically when the build is complete. `-t` can also specify different boards as well.
2. The CLI interface allows faster project initialization. Simply navigate to the desired directory and run: `pio project init -b <default board here>`
3. It allows better access to other PlatformIO options such as:



```
Usage: pio [OPTIONS] COMMAND [ARGS]...

Options:
  --version          Show the version and exit.
  -c, --caller TEXT  Caller ID (service)
  --no-ansi          Do not print ANSI control characters
  -h, --help         Show this message and exit.

Commands:
  access    Manage resource access
  account   Manage PlatformIO account
  boards    Board Explorer
  check     Static Code Analysis
  ci        Continuous Integration
  debug     Unified Debugger
  device    Device manager & Serial/Socket monitor
  home      GUI to manage PlatformIO
  org       Manage organizations
  pkg       Unified Package Manager
  project   Project Manager
  remote    Remote Development
  run       Run project targets (build, upload, clean, etc.)
  settings  Manage system settings
  system    Miscellaneous system commands
  team      Manage organization teams
  test      Unit Testing
  upgrade   Upgrade PlatformIO Core to the latest version
```