
PlatformIO Documentation

Release 3.6.0

PlatformIO

Aug 06, 2018

Contents

1	Contents	3
1.1	What is PlatformIO?	3
1.2	PlatformIO IDE	5
1.3	PlatformIO Core	7
1.4	PlatformIO Home	120
1.5	Tutorials and Examples	127
1.6	Project Configuration File <code>platformio.ini</code>	164
1.7	Environment variables	190
1.8	Advanced Scripting	193
1.9	Library Manager	200
1.10	Development Platforms	218
1.11	Frameworks	341
1.12	Embedded Boards	437
1.13	Custom Platform & Board	479
1.14	PIO Account	485
1.15	PIO Remote	486
1.16	PIO Unified Debugger	491
1.17	PIO Unit Testing	531
1.18	Cloud & Desktop IDE	537
1.19	Continuous Integration	635
1.20	Articles about us	653
1.21	Frequently Asked Questions	657
1.22	Release Notes	664
1.23	Migrating from 2.x to 3.0	688
	Bibliography	693

Cross-platform IDE and unified debugger. Remote unit testing and firmware updates.

Social: [Twitter](#) | [Facebook](#) | [Hackaday](#) | [Bintray](#) | [Community](#)

CHAPTER 1

Contents

1.1 What is PlatformIO?

Contents

- *Press about PlatformIO*
- *Awards*
- *Problematic*
- *Overview*
- *User SHOULD have a choice*
- *How does it work?*

1.1.1 Press about PlatformIO

“Different microcontrollers normally have different developing tools . For instance Arduino rely on Arduino IDE. Few more advanced users set up different graphical interfaces like Eclipse for better project management. Sometimes it may be hard to keep up with different microcontrollers and tools. You probably thought that single unified development tool could be great. Well this is what PlatformIO open source ecosystem is for.

This is cross platform code builder and library manager with platforms like Arduino or MBED support. They took care of toolchains, debuggers, frameworks that work on most popular platforms like Windows, Mac and Linux. It supports more than 200 development boards along with more than 15 development platforms and 10 frameworks. So most of popular boards are covered. They’ve done hard work in organizing and managing hundreds of libraries that can be included in to your project. Also lots of examples allow you to start developing quickly. PlatformIO initially was developed with Command line philosophy. It’s been successfully used with other IDE’s like Eclipse or Visual Studio. Recently they’ve released a version with built in IDE based on Atom text editor”, - [\[Embedds\]](#).

1.1.2 Awards

PlatformIO was nominated for the year's [best Software and Tools](#) in the 2015/16 IoT Awards.

1.1.3 Problematic

- The main problem which repulses people from embedded world is a complicated process to setup development software for a specific MCU/board: toolchains, proprietary vendor's IDE (which sometimes isn't free) and what is more, to get a computer with OS where that software is supported.
- Multiple hardware platforms (MCUs, boards) require different toolchains, IDEs, etc, and, respectively, spending time on learning new development environments.
- Finding proper libraries and code samples showing how to use popular sensors, actuators, etc.
- Sharing embedded projects between team members, regardless of operating system they prefer to work with.

1.1.4 Overview

PlatformIO is independent from the platform, in which it is running. In fact, the only requirement is Python, which exists pretty much everywhere. What this means is that PlatformIO projects can be easily moved from one computer to another, as well as that PlatformIO allows for the easy sharing of projects between team members, regardless of operating system they prefer to work with. Beyond that, PlatformIO can be run not only on commonly used desktops/laptops but also on the servers without X Window System. While PlatformIO itself is a console application, it can be used in combination with one's favorite [Cloud & Desktop IDE](#) or text editor such as [PlatformIO IDE for Atom](#), [CLion](#), [Eclipse](#), [Emacs](#), [NetBeans](#), [Qt Creator](#), [Sublime Text](#), [VIM](#), [Visual Studio](#), [PlatformIO IDE for VSCode](#), etc.

Alright, so PlatformIO can run on different operating systems. But more importantly, from development perspective at least, is a list of supported boards and MCUs. To keep things short: PlatformIO supports approximately 200 [Embedded Boards](#) and all major [Development Platforms](#).

1.1.5 User SHOULD have a choice

- Decide which operation system they want to run development process on. You can even use one OS at home and another at work.
- Choose which editor to use for writing the code. It can be pretty simple editor or powerful favorite [Cloud & Desktop IDE](#).
- Focus on the code development, significantly simplifying support for the [Development Platforms](#) and MCUs.

1.1.6 How does it work?

Without going too deep into PlatformIO implementation details, work cycle of the project developed using PlatformIO is as follows:

- Users choose board(s) interested in [Project Configuration File platformio.ini](#)
- Based on this list of boards, PlatformIO downloads required toolchains and installs them automatically.
- Users develop code and PlatformIO makes sure that it is compiled, prepared and uploaded to all the boards of interest.

1.2 PlatformIO IDE

PlatformIO IDE is the next-generation integrated development environment for IoT.

- Cross-platform build system without external dependencies to the OS software:
 - 400+ embedded boards
 - 20+ development platforms
 - 10+ frameworks
- [PIO Unified Debugger](#)
- [PIO Remote](#)
- [PIO Unit Testing](#)
- C/C++ Intelligent Code Completion
- C/C++ Smart Code Linter for rapid professional development
- Library Manager for the hundreds popular libraries
- Multi-projects workflow with multiple panes
- Themes support with dark and light colors
- Serial Port Monitor
- Built-in Terminal with [PlatformIO Core](#) and CLI tool (`pio`, `platformio`)

We provide official packages (plugins, extensions) for the most popular IDEs and text editors.

Note: In our experience, [PlatformIO IDE for VSCode](#) offers better system performance, and users have found it easier to get started

1.2.1 VSCode

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Python, PHP, Go) and runtimes (such as .NET and Unity)

```

    WiFi.cpp x Memory [0x42000800+1] ...
    579   sprintf(provSsid, "wifi101-%.2X%.");
    580
    581   // start provisioning mode
    582   startProvision(provSsid, "wifi101");
    583 }
    584
    585   return status();
    586 }
    587
    588 uint8_t WiFiClass::startProvision(const ...
    589 {
    590     strM2MAPConfig strM2MAPConfig;
    591
    592     if (!_init) {
    593         init();
    594     }
    595
    596     // Enter Provision mode:
    597     memset(&strM2MAPConfig, 0x00, sizeof(strM2MAPConfig));
    598     strcpy((char *)&strM2MAPConfig.au8SSID,
    599             "0: 192 '\300'
    600             1: 168 '\250'
    601             2: 1 '\001'
    602             3: 1 '\001'");
    603
    604     WiFiClass::startProvision@<function>
    605     WiFiClass::beginProvision@<function>
    606     WiFiClass::beginProvision@<function>
    607     setup@0x000002d6 setup...
    608     main@0x0006b82 main.d...
    609
    610     _CALL STACK PAUSED ON ST...
    611     WiFiClass::startProvision@<function>
    612     WiFiClass::beginProvision@<function>
    613     WiFiClass::beginProvision@<function>
    614     _status = WL_PROVISIONING_FAILED;
    615
    616     if (m2m_wifi_start_provision_mode((ts...
    617         _status = WL_PROVISIONING;
    618         _mode = WL_PROV_MODE;
    619
    620         memset(_ssid, 0, M2M_MAX_SSID_LEN);
    621         memcpy(_ssid, ssid, strlen(ssid));
    622         m2m_memcpy((uint8 *)&_localip, (uint8 ...
    623         _submask = 0x00FFFFFF;
    624         _gateway = _localip;
    625
    626 #ifdef CONF_PERIPH
    627         // WiFi led ON (rev A then rev B).
    628
    629 > Executing task: platformio device monitor <
    630 --- Miniterm on /dev/cu.usbmodemFA142 9600,8,N,1 ---
    631 --- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
    632 Configuring WiFi shield/module...
    633 Starting in provisioning mode...
    634
    635
    636
    637
    638
    639
    640
    641
    642
    643
  
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

> Executing task: platformio device monitor <

— Miniterm on /dev/cu.usbmodemFA142 9600,8,N,1 —

— Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H —

Configuring WiFi shield/module...

Starting in provisioning mode...

[Install PlatformIO IDE for VSCode / Get started](#)

1.2.2 Atom

Atom is a text editor that's modern, approachable, yet hackable to the core—a tool you can customize to do anything but also use productively without ever touching a config file.

The screenshot shows the PlatformIO IDE for Atom interface. On the left is a file tree with a project named 'PlatformIO' containing 'lib', 'src', and 'platformio.ini'. The 'src' folder contains 'blink.cpp'. The main editor window shows the content of 'blink.cpp' with syntax highlighting and code completion suggestions. A tooltip labeled 'Intelligent Code Completion' points to the suggestion 'Serial.begin'. Below the editor is a 'Smart Code Linter' panel showing a table of errors and warnings. The table has columns for Severity, Provider, Description, and Line. One error is listed: 'Error' provider 'GCC' description "'call_undefined' was not declared in this scope' at line 14:18". At the bottom, there are tabs for 'PIO Debug', 'src/blink.cpp*', and a status bar showing '1 1 0' errors, '15:14', and '#0 in digitalWrite() at wiring_digital.c:82'.

1.3 PlatformIO Core

PlatformIO Core (CLI tool) is a heart of whole PlatformIO ecosystem and consists of

- Multi-platform Build System
- Development platform and package managers
- *Library Manager*
- *Library Dependency Finder (LDF)*
- *Serial Port Monitor*
- Integration components (*Cloud & Desktop IDE* and *Continuous Integration*).

PlatformIO Core is written in Python 2.7 and works on Windows, macOS, Linux, FreeBSD and ARM-based credit-card sized computers (Raspberry Pi, BeagleBone, CubieBoard, Samsung ARTIK, etc.).

PlatformIO Core provides a rich and documented Command Line Interface (CLI). The other PlatformIO-based software and IDEs are based on **PlatformIO Core CLI**, such as *PlatformIO IDE*. In other words, they wrap **PlatformIO Core** with own GUI.

Note: Please note that you do not need to install **PlatformIO Core** if you are going to use *PlatformIO IDE*. **PlatformIO Core** is built into PlatformIO IDE and you will be able to use it within PlatformIO IDE Terminal.

If you need **PlatformIO Core** commands outside PlatformIO IDE, please *Install Shell Commands*.

1.3.1 Demo

Contents

- “*Blink Project*”
 - *Used in demo*
- *Platform Manager*
 - *Used in demo*
- *Library Manager*
 - *Used in demo*
- *Over-the-Air update for ESP8266*
 - *Used in demo*

“*Blink Project*”

Used in demo

1. Source code of *Wiring Blink Example*
2. *platformio run* command
3. *platformio run -t upload* command.

Platform Manager

Used in demo

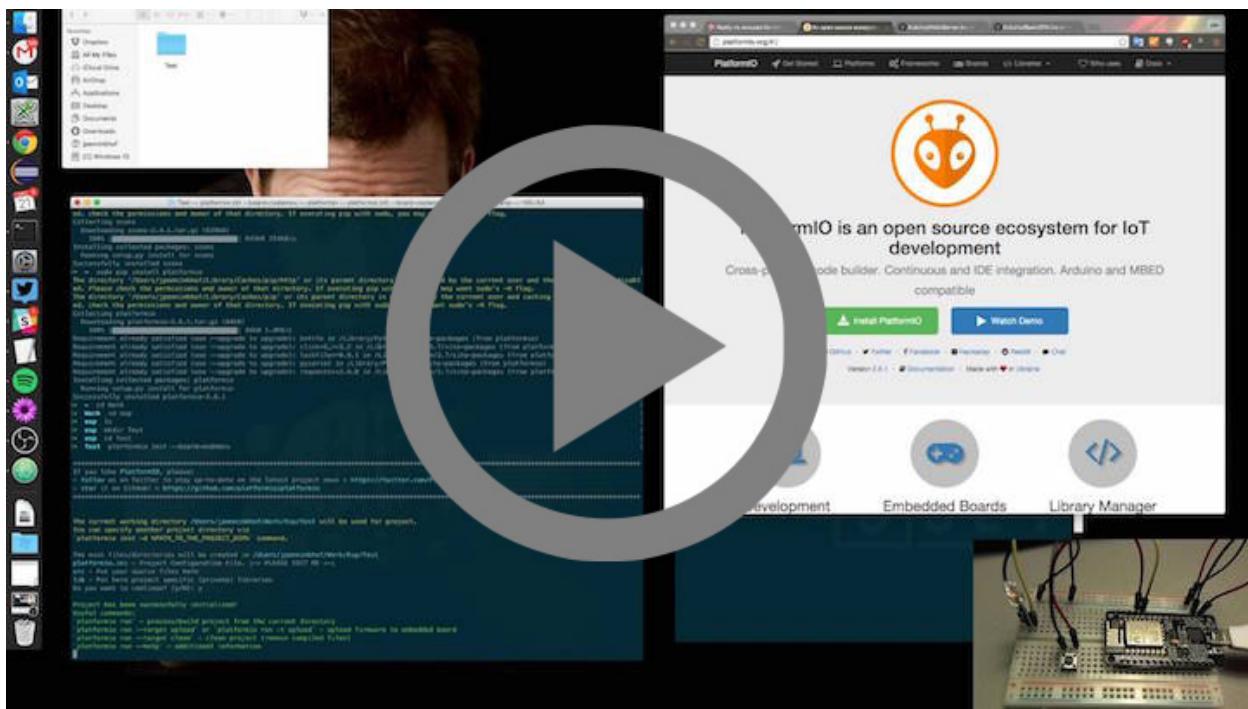
1. *Platform Manager*
2. *platformio platform list* command
3. *platformio platform search avr* command
4. *platformio platform show teensy* command
5. *platformio platform update* command.

Library Manager

Used in demo

1. *Library Manager*
2. *platformio lib search 1-wire* command
3. *platformio lib install 54* command
4. *platformio lib search -f mbed* command
5. *platformio lib search -k rf* command
6. *platformio lib search radiohead* command
7. *platformio lib install 124 –version “1.40”* command
8. *platformio lib show 124* command
9. *platformio lib update* command.

Over-the-Air update for ESP8266



Used in demo

1. *platformio run* command
2. *platformio run -t upload* command.

1.3.2 Installation

Note: Please note that you do not need to install *PlatformIO Core* if you are going to use *PlatformIO IDE*. *PlatformIO Core* is built into PlatformIO IDE and you will be able to use it within PlatformIO IDE Terminal.

If you need *PlatformIO Core* outside PlatformIO IDE, please [Install Shell Commands](#).

PlatformIO Core is written in [Python 2.7](#) and works on Windows, macOS, Linux, FreeBSD and ARM-based credit-card sized computers ([Raspberry Pi](#), [BeagleBone](#), [CubieBoard](#), [Samsung ARTIK](#), etc.).

- [System requirements](#)
- [Installation Methods](#)
 - [Python Package Manager](#)
 - [Installer Script](#)
 - * [Super-Quick \(Mac / Linux\)](#)
 - * [Local Download \(Mac / Linux / Windows\)](#)
 - [macOS Homebrew](#)
 - [Full Guide](#)
 - [Virtual Environment](#)
 - * [Prerequisites](#)
 - * [Creating](#)
- [Development Version](#)
- [Install Shell Commands](#)
 - [Unix and Unix-like](#)
 - * [Method 1](#)
 - * [Method 2](#)
 - * [Method 3](#)
 - [Windows](#)
- [Uninstall PIO Core and dependent packages](#)
- [Troubleshooting](#)

System requirements

Operating System Windows, macOS, Linux, FreeBSD, Linux ARMv6+

Python Interpreter Python 2.7 is required. PlatformIO **does not** support Python 3. See detailed instruction how to [Install Python Interpreter](#) for Windows.

Terminal Application All commands below should be executed in [Command-line](#) application (Terminal). For macOS and Linux OS - *Terminal* application, for Windows OS – cmd.exe application.

Access to Serial Ports (USB/UART) Windows Users: Please check that you have correctly installed USB driver from board manufacturer

Linux Users:

- Ubuntu/Debian users may need to add own “username” to the “dialout” group if they are not “root”, doing this issuing a sudo usermod -a -G dialout yourusername.
- Install “udev” rules file [99-platformio-udev.rules](#) (an instruction is located in the file).
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Installation Methods

Please *choose ONE* of the following methods:

- [Python Package Manager](#)
- [Installer Script](#)
 - [Super-Quick \(Mac / Linux\)](#)
 - [Local Download \(Mac / Linux / Windows\)](#)
- [macOS Homebrew](#)
- [Full Guide](#)
- [Virtual Environment](#)
 - [Prerequisites](#)
 - [Creating](#)

Python Package Manager

The latest stable version of PlatformIO may be installed or upgraded via Python Package Manager ([pip](#)) as follows:

```
pip install -U platformio
```

If `pip` command is not available run `easy_install pip` or use [Installer Script](#) which will install `pip` and `platformio` automatically.

Note that you may run into permissions issues running these commands. You have a few options here:

- Run with `sudo` to install PlatformIO and dependencies globally
- Specify the `pip install --user` option to install local to your user
- Run the command in a `virtualenv` local to a specific project working set.

Installer Script

Super-Quick (Mac / Linux)

To install or upgrade *PlatformIO* paste that at a *Terminal* prompt (**MAY require** administrator access `sudo`):

```
python -c "$(curl -fsSL https://raw.githubusercontent.com/platformio/platformio/develop/scripts/get-platformio.py)"
```

Local Download (Mac / Linux / Windows)

To install or upgrade *PlatformIO*, download (save as...) `get-platformio.py` script. Then run the following (**MAY require** administrator access `sudo`):

```
# change directory to folder where is located downloaded "get-platformio.py"  
cd /path/to/dir/where/is/located/get-platformio.py/script  
  
# run it  
python get-platformio.py
```

On *Windows OS* it may look like:

```
# change directory to folder where is located downloaded "get-platformio.py"  
cd C:\path\to\dir\where\is\located\get-platformio.py\script  
  
# run it  
C:\Python27\python.exe get-platformio.py
```

macOS Homebrew

The latest stable version of PlatformIO may be installed or upgraded via macOS Homebrew Packages Manager (`brew`) as follows:

```
brew install platformio
```

Full Guide

1. Check a python version (only Python 2.7 is supported):

```
python --version
```

Windows Users only:

- Download [Python 2.7](#) and install it.
- Add to PATH system variable ;`C:\Python27`; `C:\Python27\Scripts`; and reopen *Command Prompt* (`cmd.exe`) application. Please read this article [How to set the path and environment variables in Windows](#).

2. Install a `platformio` and related packages:

```
pip install -U platformio
```

If your computer does not recognize `pip` command, try to install it first using [these instructions](#).

For upgrading `platformio` to the latest version:

```
pip install -U platformio
```

Virtual Environment

PlatformIO Core may be installed into isolated Python environment. This method is very good if you don't want to install PlatformIO Core Python's dependencies (packages) into your global system scope. [PlatformIO IDE](#) uses this method to install PlatformIO Core.

Default and recommended environment folder is “`home_dir`/penv”. You can print **environment folder path** using the next command in your system terminal:

```
python -c_
→"import os; print os.path.join(os.getenv('PLATFORMIO_HOME_DIR', os.path.expanduser('~')),"penv")"

#####
##### Examples
# Windows
# C:\Users\UserName\.platformio\penv

# Linux
# ~/.platformio/penv
# /home/username/.platformio/penv

# macOS
# ~/.platformio/penv
# /Users/username/.platformio/penv
```

Prerequisites

1. Please remove existing PlatformIO Core **environment folder** if exists. See above how to get a path.
2. Please check that you have a valid Python interpreter running a next command in system terminal. Python 2.7.9+ is recommended.

```
python --version

# or, for Unix (Linux, Mac), you can use `python2` or `python2.7` aliases
python2 --version
python2.7 --version
```

Warning: Windows Users: If you already tried to install [PlatformIO IDE](#) and did not get success, please open system's Control Panel > Installed Programs, and check if PlatformIO IDE tried to install an own isolated Python 2.7 version. Please uninstall it. Also is good to uninstall all Python interpreters from a system and install manually the latest Python 2.7 using [Install Python Interpreter](#) guide.

Please note, that you can have Python 3 installed in a system too. You will need to specify full path to Python 2.7 when creating a virtual environment (explained below).

3. Make sure `virtualenv --help` command exists in a system, otherwise, please install it manually using `pip install virtualenv` or `pip2 install virtualenv` command.
If `pip` (Python Package Manager) does not exists, you have to install it manually. See <https://pip.pypa.io/en/stable/installing/>

Creating

1. Create a folder which contains all the necessary executables to use the packages that PIO Core would need using `virtualenv` command:

```
virtualenv /path/to/.platformio/penv

# if you have multiple Python interpreters in a system, please specify
# a valid Python 2.7 via ``-p, --python`` option:
virtualenv -p python2 /path/to/.platformio/penv

# or using full path to Python 2.7 interpreter
virtualenv --python=/path/to/python2.7 /path/to/.platformio/penv

# EXAMPLES
# Windows
virtualenv C:\Users\UserName\.platformio\penv
virtualenv --python=C:\Python27\python.exe C:\Users\UserName\.platformio\penv

# Unix (Linux, Mac)
virtualenv ~/.platformio/penv
virtualenv -p python2.7 ~/.platformio/penv
```

2. Activate virtual environment

```
# Windows
/path/to/.platformio/penv/Scripts/activate

# Unix (Linux, Mac)
/path/to/.platformio/penv/bin/activate
```

3. Install PIO Core into virtual environment

```
pip install -U platformio
```

If you plan to use PIO Core commands outside virtual environment, please [Install Shell Commands](#).

Development Version

Warning: If you use *PlatformIO IDE*, please enable development version:

- **Atom:** “Menu PlatformIO: Settings > PlatformIO IDE > Use development version of PlatformIO Core”
- **VSCode:** Set `platformio-ide.useDevelopmentPIOCore` to true in Settings.

Install the latest PlatformIO from the develop branch:

```
# uninstall existing version
pip uninstall platformio

# install the latest development version of PlatformIO
pip install -U https://github.com/platformio/platformio-core/archive/develop.zip
```

If you want to be up-to-date with the latest develop version of PlatformIO, then you need to re-install PlatformIO each time if you see the new commits in [PlatformIO GitHub repository](#) (branch: develop).

To revert to the latest stable version

```
pip uninstall platformio
pip install -U platformio
```

Install Shell Commands

PlatformIO Core consists of 2 standalone tools in a system:

- `platformio` or `pio` (short alias) - [User Guide](#)
- `piodebuggdb` - alias of `platformio debug`

If you have *PlatformIO IDE* already installed, you do not need to install *PlatformIO Core* separately. Just link these tools with your shell:

- *Unix and Unix-like*
 - [Method 1](#)
 - [Method 2](#)
 - [Method 3](#)
- *Windows*

Unix and Unix-like

In Unix and Unix-like systems, there are multiple ways to achieve this.

Method 1

You can export PlatformIO executables' directory to the PATH environmental variable. This method will allow you to execute `platformio` commands from any terminal emulator as long as you're logged in as the user PlatformIO is installed and configured for.

If you use Bash as your default shell, you can do it by editing either `~/.profile` or `~/.bash_profile` and adding the following line:

```
export PATH=$PATH:~/platformio/penv/bin
```

If you use Zsh, you can either edit `~/.zprofile` and add the code above, or for supporting both, Bash and Zsh, you can first edit `~/.profile` and add the code above, then edit `~/.zprofile` and add the following line:

```
emulate sh -c '. ~/.profile'
```

After everything's done, just restart your session (log out and log back in) and you're good to go.

If you don't know the difference between the two, check out [this page](#).

Method 2

Go to the *PlatformIO* menu → *Settings* → *PlatformIO IDE*, scroll down to the *Custom PATH for 'platformio' command* and enter the following: `~/platformio/penv/bin`. After you've done that, you'll need to go to the

PlatformIO menu → *Settings* → *PlatformIO IDE Terminal*, scroll down to the *Toggles* section and uncheck the *Login Shell* checkbox. Finally, restart Atom and check out the result.

Method 3

You can create system-wide symlinks. This method is not recommended if you have multiple users on your computer because the symlinks will be broken for other users and they will get errors while executing PlatformIO commands. If that's not a problem, open your system terminal app and paste these commands (**MAY require** administrator access sudo):

```
ln -s ~/.platformio/penv/bin/platformio /usr/local/bin/platformio
ln -s ~/.platformio/penv/bin/pio /usr/local/bin/pio
ln -s ~/.platformio/penv/bin/piodebugdb /usr/local/bin/piodebugdb
```

After that, you should be able to run PlatformIO from terminal. No restart is required.

Windows

Please read one of these instructions [How do I set or change the PATH system variable?](#)

You need to edit system environment variable called Path and append C:\Users\UserName\.platformio\env\Scripts; path in the beginning of a list (please replace UserName with your account name).

Uninstall PIO Core and dependent packages

- Uninstall PIO Core tool

```
# uninstall standalone PIO Core installed via `pip`
pip uninstall platformio

# uninstall Homebrew's PIO Core (only macOS users if you installed it via
# Homebrew before)
brew uninstall platformio
```

- Dependent packages, global libraries are installed to [home_dir](#) folder (in user's HOME directory). Just remove it.

Troubleshooting

Note: **Linux OS:** Don't forget to install "udev" rules file [99-platformio-udev.rules](#) (an instruction is located in the file).

Windows OS: Please check that you have correctly installed USB driver from board manufacturer

For further details, frequently questions, known issues, please refer to [Frequently Asked Questions](#).

1.3.3 Quick Start

This tutorial introduces you to the basics of [PlatformIO Core](#) Command Line Interface (CLI) workflow and shows you a creation process of a simple cross-platform “Blink” Project. After finishing you will have a general understanding of how to work with the multiple development platforms and embedded boards.

Setting Up the Project

[PlatformIO Core](#) provides special `platformio init` command for configuring your projects. It allows to initialize new empty project or update existing with the new data.

What is more, `platformio init` can be used for [Cloud & Desktop IDE](#). It means that you will be able to import pre-generated PlatformIO project using favorite IDE and extend it with the professional instruments for IoT development.

This tutorial is based on the next popular embedded boards and development platforms using [Arduino Wiring-based Framework](#):

Platform	Board	Framework
Atmel AVR	Arduino Uno (8-bit ATmega328P)	Arduino Wiring-based Framework
Espressif	NodeMCU 1.0 (32-bit ESP8266)	Arduino Wiring-based Framework
Teensy	Teensy 3.1 (32-bit ARM MK20DX256)	Arduino Wiring-based Framework

Board Identifier

`platformio init` command requires to specify board identifier (ID/TYPE). It can be found using [Embedded Boards Explorer](#) or `platformio boards` command. For example, using `platformio boards` let's try to find Teensy boards:

```
> platformio boards teensy

Platform: teensy
-----
Type          MCU        Frequency   Flash     RAM      Name
-----
teensy20      atmega32u4  16MHz      31K      2.5K    Teensy 2.0
teensy30      mk20dx128  48MHz      128K     16K     Teensy 3.0
teensy31      mk20dx256  72MHz      256K     64K     Teensy 3.1 / 3.2
teensylc     mk126z64   48MHz      62K      8K      Teensy LC
teensy20pp    at90usb1286 16MHz      127K     8K      Teensy++ 2.0
```

According to the table above the ID/TYPER for Teensy 3.1 is `teensy31`. Also, the ID for Arduino UNO is `uno` and for NodeMCU 1.0 (ESP-12E Module) is `nodemcuv2`.

Initialize Project

PlatformIO ecosystem contains big database with pre-configured settings for the most popular embedded boards. It helps you to forget about installing toolchains, writing build scripts or configuring uploading process. Just tell PlatformIO the Board ID and you will receive full working project with pre-installed instruments for the professional development.

1. Create empty folder where you are going to initialize new PlatformIO project. Then open system Terminal and change directory to it:

```
# create new direcotry  
> mkdir path_to_the_new_directory  
  
# go to it  
> cd path_to_the_new_directory
```

2. Initialize project for the boards mentioned above (you can specify more than one board at time):

```
> platformio init --board uno --board nodemcuv2 --board teensy31  
  
The current working directory *** will be used for the new project.  
You can specify another project directory via  
'platformio init -d %PATH_TO_THE_PROJECT_DIR%' command.  
  
The next files/directories will be created in ***  
platformio.ini - Project Configuration File. |-> PLEASE EDIT ME <-|  
src - Put your source files here  
lib - Put here project specific (private) libraries  
Do you want to continue? [y/N]: y  
Project has been successfully initialized!  
Useful commands:  
'platformio run' - process/build project from the current directory  
'platformio run --target upload' or 'platformio run -t upload' - upload firmware  
to embedded board  
'platformio run --target clean' - clean project (remove compiled files)
```

Congrats! You have just created the first PlatformIO based Project with the next structure:

- *Project Configuration File* `platformio.ini`
- `src` directory where you should place source code (`*.h`, `*.c`, `*.cpp`, `*.S`, `*.ino`, etc.)
- `lib` directory can be used for the project specific (private) libraries. More details are located in `lib/readme.txt` file.
- Miscellaneous files for VCS and *Continuous Integration* support.

Note: If you need to add new board to the existing project please use `platformio init` again.

The result of just generated `platformio.ini`:

```
; PlatformIO Project Configuration File  
;  
; Build options: build flags, source filter, extra scripting  
; Upload options: custom port, speed and extra flags  
; Library options: dependencies, extra library storages  
;  
; Please visit documentation for the other options and examples  
; http://docs.platformio.org/page/projectconf.html  
  
[env:uno]  
platform = atmelavr  
framework = arduino  
board = uno  
  
[env:nodemcuv2]  
platform = espressif8266
```

(continues on next page)

(continued from previous page)

```
framework = arduino
board = nodemcu2

[env:teensy31]
platform = teensy
framework = arduino
board = teensy31
```

Now, we need to create `main.cpp` file and place it to `src` folder of our newly created project. The contents of `src/main.cpp`:

```
/***
 * Blink
 *
 * Turns on an LED on for one second,
 * then off for one second, repeatedly.
 */
#include "Arduino.h"

#ifndef LED_BUILTIN
#define LED_BUILTIN 13
#endif

void setup()
{
    // initialize LED digital pin as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
    // turn the LED on (HIGH is the voltage level)
    digitalWrite(LED_BUILTIN, HIGH);

    // wait for a second
    delay(1000);

    // turn the LED off by making the voltage LOW
    digitalWrite(LED_BUILTIN, LOW);

    // wait for a second
    delay(1000);
}
```

The final Project structure:

```
project_dir
└── lib
    └── readme.txt
── platformio.ini
└── src
    └── main.cpp
```

Process Project

PlatformIO Core provides special `platformio run` command to process project. If you call it without any arguments,

PlatformIO Build System will process all project environments (which were created per each board specified above). Here are a few useful commands:

- `platformio run`. Process (build) all environments specified in *Project Configuration File platformio.ini*
- `platformio run --target upload`. Build project and upload firmware to the all devices specified in *Project Configuration File platformio.ini*
- `platformio run --target clean`. Clean project (delete compiled objects)
- `platformio run -e uno`. Process only uno environment
- `platformio run -e uno -t upload`. Build project only for uno and upload firmware.

Please follow to `platformio run --target` documentation for the other targets.

Finally, demo which demonstrates building project and uploading firmware to Arduino Uno:

Further Reading

- [Project examples](#)
- [User Guide](#) for *PlatformIO Core* commands

1.3.4 User Guide

Contents

- [User Guide](#)
 - [Usage](#)
 - [Options](#)
 - [Commands](#)

Usage

```
pio [OPTIONS] COMMAND
platformio [OPTIONS] COMMAND

# "pio" is the alias of "platformio" command
```

Options

--force, -f

Force to accept any confirmation prompts and disable progress bars.

--version

Show the version of PlatformIO

--help, -h

Show help for the available options and commands

```
$ platformio --help  
$ platformio COMMAND --help
```

Commands

platformio account

Helper command for *PIO Account*.

```
# Create PIO Account  
pio account register  
  
# Login with credentials (will be sent to your e-mail)  
pio account login  
  
# Change temporary password (from e-mail) to permanent  
pio account password
```

To print all available commands and options use:

```
pio account --help  
platformio account --help  
platformio account COMMAND --help
```

platformio account forgot

Contents

- *platformio account forgot*
 - *Usage*
 - *Description*
 - * *Options*

Usage

```
platformio account forgot [OPTIONS]  
pio account forgot [OPTIONS]
```

Description

Allows you to reset password for *PIO Account* using E-Mail that was specified for registration.

Options

--username, -u

User name (E-Mail). You can omit this option and enter E-Mail in Forgot Wizard later.

platformio account login

Contents

- *platformio account login*
 - *Usage*
 - *Description*
 - * *Options*

Usage

```
platformio account login [OPTIONS]
pio account login [OPTIONS]
```

Description

Log in to *PIO Account*. If you are not able to provide authentication credentials manually you can use *PLATFORMIO_AUTH_TOKEN*. This is very useful for *Continuous Integration* systems and *PIO Remote* operations .

Options

--username, -u

User name (E-Mail). You can omit this option and enter E-Mail in Login Wizard later.

--password, -p

You can omit this option and enter securely password in Login Wizard later.

platformio account logout

Contents

- *platformio account logout*
 - *Usage*
 - *Description*

Usage

```
platformio account logout  
pio account logout
```

Description

Log out of *PIO Account*.

platformio account password

Contents

- *platformio account password*
 - *Usage*
 - *Description*

Usage

```
platformio account password  
pio account password
```

Description

Change password for *PIO Account*.

platformio account register

Contents

- *platformio account register*
 - *Usage*
 - *Description*
 - * *Options*

Usage

```
platformio account register [OPTIONS]  
pio account register [OPTIONS]
```

Description

Create a new *PIO Account*. A registration is FREE.

Options

--username, -u

User name (E-Mail). You can omit this option and enter E-Mail in Register Wizard later.

platformio account show

Contents

- *platformio account show*
 - *Usage*
 - *Description*
 - * *Options*

Usage

```
platformio account show
pio account show
```

Description

Show detailed information about *PIO Account*:

- Active groups and expiration
- Group permissions

Options

--json-output

Return the output in **JSON** format

platformio account token

Contents

- *platformio account token*

- *Usage*
- *Description*
- * *Options*

Usage

```
platformio account token
pio account token
```

Description

Get or regenerate Personal Authentication Token. It is very useful for *Continuous Integration* systems, *PIO Remote* operations where you are not able to authorize manually.

PlatformIO handles Personal Authentication Token from environment variable `PLATFORMIO_AUTH_TOKEN`.

Options

--regenerate

If this option is specified a new authentication token will be generated.

--json-output

Return the output in `JSON` format

platformio boards

Contents

- *platformio boards*
 - *Usage*
 - *Description*
 - * *Options*
 - *Examples*

Usage

```
platformio boards [OPTIONS] [FILTER]
pio boards [OPTIONS] [FILTER]
```

Description

List pre-configured Embedded Boards

Options

--installed

List boards only from the installed platforms

--json-output

Return the output in **JSON** format

Examples

1. Show all available pre-configured embedded boards

```
$ platformio boards

Platform: atmelavr
-----
Type          MCU        Frequency  Flash   RAM    Name
-----
btatmega168   atmega168  16MHz     14K    1K    Arduino BT ATmega168
btatmega328   atmega328p 16MHz     28K    2K    Arduino BT ATmega328
diecimilaatmega168 atmega168  16MHz     14K    1K    Arduino Duemilanove or_
 ↵Diecimila ATmega168
diecimilaatmega328 atmega328p 16MHz     30K    2K    Arduino Duemilanove or_
 ↵Diecimila ATmega328
esplora        atmega32u4  16MHz     28K    2K    Arduino Esplora
ethernet       atmega328p  16MHz     31K    2K    Arduino Ethernet
...

```

2. Filter Arduino-based boards

```
$ platformio boards arduino

Platform: atmelavr
-----
Type          MCU        Frequency  Flash   RAM    Name
-----
btatmega168   atmega168  16MHz     14K    1K    Arduino BT ATmega168
btatmega328   atmega328p 16MHz     28K    2K    Arduino BT ATmega328
diecimilaatmega168 atmega168  16MHz     14K    1K    Arduino Duemilanove or_
 ↵Diecimila ATmega168
diecimilaatmega328 atmega328p 16MHz     30K    2K    Arduino Duemilanove or_
 ↵Diecimila ATmega328
esplora        atmega32u4  16MHz     28K    2K    Arduino Esplora
ethernet       atmega328p  16MHz     31K    2K    Arduino Ethernet
...

```

3. Filter mbed-enabled boards

```
$ platformio boards mbed

Platform: freescalekinetis
-----
Type          MCU        Frequency  Flash   RAM    Name
-----
frdm_k20d50m  mk20dx128vlh5 48MHz    128K   16K    Freescale Kinetis FRDM-
 ↵K20D50M

```

(continues on next page)

(continued from previous page)

frdm_k22f	mk22fn512vlh12	120MHz	512K	128K	Freescale Kinetis FRDM-
↳ K22F					
...					
Platform: nordicnrf51					
Type	MCU	Frequency	Flash	RAM	Name
wallBotBLE	nrf51822	16MHz	128K	16K	JKSoft Wallbot BLE
nrf51_dk	nrf51822	32MHz	256K	32K	Nordic nRF51-DK
...					
Platform: nxplpc					
Type	MCU	Frequency	Flash	RAM	Name
blueboard_lpc11u24	lpc11u24	48MHz	32K	8K	BlueBoard-LPC11U24
dipcortexm0	lpc11u24	50MHz	32K	8K	DipCortex M0
lpc11u35	lpc11u35	48MHz	64K	10K	EA LPC11U35 QuickStart
↳ Board					
...					
Platform: ststm32					
Type	MCU	Frequency	Flash	RAM	Name
disco_f401vc	stm32f401vct6	84MHz	256K	64K	32F401CDISCOVERY
nucleo_f030r8	stm32f030r8t6	48MHz	64K	8K	ST Nucleo F030R8
...					

4. Filter boards which are based on ATmega168 MCU

\$ platformio boards atmega168					
Platform: atmelavr					
Type	MCU	Frequency	Flash	RAM	Name
btagtmega168	atmega168	16MHz	14K	1K	Arduino BT ATmega168
diecimilaatmega168	atmega168	16MHz	14K	1K	Arduino Duemilanove or
↳ Diecimila ATmega168					
miniatmega168	atmega168	16MHz	14K	1K	Arduino Mini ATmega168
atmegangatmega168	atmega168	16MHz	14K	1K	Arduino NG or older
↳ ATmega168					
nanoatmega168	atmega168	16MHz	14K	1K	Arduino Nano ATmega168
pro8MHzatmega168	atmega168	8MHz	14K	1K	Arduino Pro or Pro Mini
↳ ATmega168 (3.3V, 8 MHz)					
pro16MHzatmega168	atmega168	16MHz	14K	1K	Arduino Pro or Pro Mini
↳ ATmega168 (5V, 16 MHz)					
lilypadatmega168	atmega168	8MHz	14K	1K	LilyPad Arduino ATmega168
168pa16m	atmega168p	16MHz	15K	1K	Microduino Core
↳ (Atmega168PA@16M, 5V)					
168pa8m	atmega168p	8MHz	15K	1K	Microduino Core
↳ (Atmega168PA@8M, 3.3V)					

5. Show boards by TI MSP430

```
$ platformio boards timsp430

Platform: timsp430
-----
Type          MCU      Frequency  Flash   RAM    Name
-----
lpmsp430fr5739  msp430fr5739  16MHz    15K    1K    FraunchPad w/ msp430fr5739
lpmsp430f5529   msp430f5529   16MHz    128K   1K    LaunchPad w/ msp430f5529_
 ↪ (16MHz)
lpmsp430f5529_25  msp430f5529  25MHz    128K   1K    LaunchPad w/ msp430f5529_
 ↪ (25MHz)
lpmsp430fr5969   msp430fr5969  8MHz     64K    1K    LaunchPad w/ msp430fr5969
lpmsp430g2231    msp430g2231   1MHz     2K     128B   LaunchPad w/ msp430g2231_
 ↪ (1MHz)
lpmsp430g2452    msp430g2452   16MHz    8K     256B   LaunchPad w/ msp430g2452_
 ↪ (16MHz)
lpmsp430g2553    msp430g2553   16MHz    16K    512B   LaunchPad w/ msp430g2553_
 ↪ (16MHz)
```

platformio ci

Contents

- *platformio ci*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio ci [OPTIONS] [SRC]
pio ci [OPTIONS] [SRC]
```

Description

platformio ci command is conceived of as “hot key” for building project with arbitrary source code structure. In a nutshell, using SRC and *platformio ci --lib* contents PlatformIO initializes via *platformio init* new project in *platformio ci --build-dir* with the build environments (using *platformio ci --board* or *platformio ci --project-conf*) and processes them via *platformio run* command.

platformio ci command accepts **multiple** SRC arguments, *platformio ci --lib* and *platformio ci --exclude* options which can be a path to directory, file or **Glob Pattern**. Also, you can omit SRC argument and set path (multiple paths are allowed denoting with :) to **PLATFORMIO_CI_SRC** Environment variable

For more details as for integration with the popular Continuous Integration Systems please follow to *Continuous Integration* page.

Note: `platformio ci` command is useful for library developers. It allows to build different examples without creating own project per them. Also, is possible to upload firmware to the target device. In this case, you need to pass additional option `--project-option="targets=upload"`. What is more, you can specify custom upload port using `--project-option="upload_port=<port>"` option. See `platformio ci --project-option` for details.

Options

-l, --lib

Source code which will be copied to `<BUILD_DIR>/lib` directly.

If `platformio ci --lib` is a path to file (not to directory), then PlatformIO will create temporary directory within `<BUILD_DIR>/lib` and copy the rest files into it.

--exclude

Exclude directories and/or files from `platformio ci --build-dir`. The path must be relative to PlatformIO project within `platformio ci --build-dir`.

For example, exclude from project `src` directory:

- examples folder
- *.h files from foo folder

```
platformio ci --exclude=src/examples --exclude=src/foo/*.h [SRC]
```

-b, --board

Build project with automatically pre-generated environments based on board settings.

For more details please look into `platformio init --board`.

--build-dir

Path to directory where PlatformIO will initialise new project. By default it's temporary directory within your operation system.

Note: This directory will be removed at the end of build process. If you want to keep it, please use `platformio ci --keep-build-dir`.

--keep-build-dir

Don't remove `platformio ci --build-dir` after build process.

-P, --project-conf

Buid project using pre-configured *Project Configuration File* `platformio.ini`.

-O, --project-option

Pass additional options from *Project Configuration File* `platformio.ini` to `platformio init` command. Use multiple `--project-option` options to pass multiple options to `platformio init` command. For example, automatically install dependent libraries `platformio ci --project-option="lib_deps=ArduinoJSON"` or ignore specific library `platformio ci --project-option="lib_ignore=SomeLib"`.

-v, --verbose

Shows detailed information when processing environments.

This option can be set globally using `force_verbose` setting or by environment variable `PLATFORMIO_SETTING_FORCE_VERBOSE`.

Examples

For the others examples please follow to [Continuous Integration](#) page.

platformio debug

Helper command for [PIO Unified Debugger](#).

Contents

- *platformio debug*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio debug [OPTIONS]
pio debug [OPTIONS]

# A binary shortcut for "platformio debug --interface=gdb" command
piodebuggdb [GDB OPTIONS]
```

Description

Prepare PlatformIO project for debugging or launch debug server.

Options

-e, --environment

Debug specified environments.

You can also specify which environments should be used for debugging by default using `env_default` option from [Project Configuration File platformio.ini](#).

-d, --project-dir

Specify the path to a project directory. By default, `--project-dir` is equal to a current working directory (CWD).

--interface

PIO Debugging Interface. Valid values:

- gdb - GDB: The GNU Project Debugger

-v, --verbose

Shows detailed information when processing environments.

This option can be set globally using `force_verbose` setting or by environment variable `PLATFORMIO_SETTING_FORCE_VERBOSE`.

Examples

1. Prepare a project for debugging

```
> platformio debug

[Sun Apr 30 01:34:01 2017] Processing mzeropro (platform: atmelsam; debug_extra_cmds:_
↳ b main.cpp:26; board: mzeropro; framework: arduino)
-----
→-----  

Verbose mode can be enabled via `'-v, --verbose` option
Collected 26 compatible libraries
Looking for dependencies...
Project does not have dependencies
Compiling .pioenvs/mzeropro/src/main.o
Compiling .pioenvs/mzeropro/FrameworkArduinoVariant/variant.o
Compiling .pioenvs/mzeropro/FrameworkArduino/IPAddress.o
Compiling .pioenvs/mzeropro/FrameworkArduino/Print.o
Archiving .pioenvs/mzeropro/libFrameworkArduinoVariant.a
Indexing .pioenvs/mzeropro/libFrameworkArduinoVariant.a
...
Compiling .pioenvs/mzeropro/FrameworkArduino/wiring_analog.o
Compiling .pioenvs/mzeropro/FrameworkArduino/wiring_digital.o
Compiling .pioenvs/mzeropro/FrameworkArduino/wiring_private.o
Compiling .pioenvs/mzeropro/FrameworkArduino/wiring_shift.o
Archiving .pioenvs/mzeropro/libFrameworkArduino.a
Indexing .pioenvs/mzeropro/libFrameworkArduino.a
Linking .pioenvs/mzeropro/firmware.elf
Calculating size .pioenvs/mzeropro/firmware.elf
Building .pioenvs/mzeropro/firmware.bin
text      data      bss      dec      hex filename
11512      256     1788    13556    34f4 .pioenvs/mzeropro/firmware.elf
===== [SUCCESS] Took 7.82 seconds =====
```

2. Launch GDB instance and load initial configuration per a project

```
> platformio debug --interface=gdb -x .pioinit

PIO Plus (https://pioplus.com) v0.8.2
...
Loading section .text, size 0x2c98 lma 0x4000
Loading section .ramfunc, size 0x60 lma 0x6c98
Loading section .data, size 0x100 lma 0x6cf8
Start address 0x47b0, load size 11768
Transfer rate: 4 KB/sec, 3922 bytes/write.
target halted due to debug-request, current mode: Thread
xPSR: 0x81000000 pc: 0x000028f4 msp: 0x20002c00
target halted due to debug-request, current mode: Thread
```

(continues on next page)

(continued from previous page)

```
xPSR: 0x81000000 pc: 0x000028f4 msp: 0x20002c00
Breakpoint 2 at 0x413a: file src/main.cpp, line 26.
```

platformio device

Contents

- *platformio device*
 - *platformio device list*
 - * *Usage*
 - * *Description*
 - * *Options*
 - * *Examples*
 - *platformio device monitor*
 - * *Usage*
 - * *Description*
 - * *Options*
 - * *Examples*

platformio device list

Usage

```
platformio device list [OPTIONS]
pio device list [OPTIONS]
```

Description

List available devices. Default is set to `--serial` and all available [Serial Ports](#) will be shown.

Options

--serial

List available Serial Ports, default.

--logical

List available logical devices.

--mdns

List multicast DNS services.

--json-output

Return the output in **JSON** format.

Examples**1. Unix OS**

```
$ platformio device list
/dev/cu.SLAB_USBtoUART
-----
Hardware ID: USB VID:PID=10c4:ea60 SNR=0001
Description: CP2102 USB to UART Bridge Controller

/dev/cu.uart-1CFF4676258F4543
-----
Hardware ID: USB VID:PID=451:f432 SNR=1CFF4676258F4543
Description: Texas Instruments MSP-FET430UIF
```

2. Windows OS

```
$ platformio device list
COM4
-----
Hardware ID: USB VID:PID=0451:F432
Description: MSP430 Application UART (COM4)

COM3
-----
Hardware ID: USB VID:PID=10C4:EA60 SNR=0001
Description: Silicon Labs CP210x USB to UART Bridge (COM3)
```

3. List multicast DNS services and logical devices

```
$ platformio device list --mdns --logical
Multicast DNS Services
=====
PlatformIO._bttremote._tcp.local.
-----
Type: _bttremote._tcp.local.
IP: ...
Port: 62941
Properties: ...

Time for PlatformIO._adisk._tcp.local.
-----
Type: _adisk._tcp.local.
IP: 192.168.0.1
Port: 9
Properties: ...

PlatformIO._ssh._tcp.local.
-----
Type: _ssh._tcp.local.
IP: ...
Port: 22
```

(continues on next page)

(continued from previous page)

```
PlatformIO._sftp-ssh._tcp.local.  
-----  
Type: _sftp-ssh._tcp.local.  
IP: ...  
Port: 22  
  
Logical Devices  
=====  
/  
-  
Name:  
  
/Volumes/PIO  
-----  
Name: PIO  
  
/Volumes/PLUS  
-----  
Name: PLUS
```

platformio device monitor

Usage

```
platformio device monitor [OPTIONS]
```

Description

This is a console application that provides a small terminal application. It is based on [Miniterm](#) and itself does not implement any terminal features such as *VT102* compatibility. However it inherits these features from the terminal it is run. For example on GNU/Linux running from an *xterm* it will support the escape sequences of the *xterm*. On Windows the typical console window is dumb and does not support any escapes. When *ANSI.sys* is loaded it supports some escapes.

To control *monitor* please use these “hot keys”:

- Ctrl+C Quit
- Ctrl+T Menu
- Ctrl+T followed by Ctrl+H Help

Options

-p, --port

Port, a number or a device name.

Could be customized in *Project Configuration File platformio.ini* using [*monitor_port*](#) option.

-b, --baud

Set baud rate, default 9600.

Could be customized in *Project Configuration File platformio.ini* using `monitor_speed` option.

--parity

Set parity (*None, Even, Odd, Space, Mark*), one of [N, E, O, S, M], default N

--rtscts

Enable RTS/CTS flow control, default Off

--xonxoff

Enable software flow control, default Off

--rts

Set initial RTS line state (0 or 1).

Could be customized in *Project Configuration File platformio.ini* using `monitor_rts` option.

--dtr

Set initial DTR line state (0 or 1).

Could be customized in *Project Configuration File platformio.ini* using `monitor_dtr` option.

--echo

Enable local echo, default Off

--encoding

Set the encoding for the serial port (e.g. hexlify, Latin1, UTF-8), default UTF-8.

-f, --filter

Add text transformation. Available filters:

- `colorize` Apply different colors for received and echo
- `debug` Print what is sent and received
- `default` Remove typical terminal control codes from input
- `direct` Do-nothing: forward all data unchanged
- `nocontrol` Remove all control codes, incl. CR+LF
- `printable` Show decimal code for all non-ASCII characters and replace most control codes

--eol

End of line mode (CR, LF or CRLF), default CRLF

NEW: Available in Miniterm/PySerial 3.0

--raw

Do not apply any encodings/transformations

--exit-char

ASCII code of special character that is used to exit the application, default 3 (DEC, Ctrl+C).

For example, to use Ctrl+] run `platformio device monitor --exit-char 29`.

--menu-char

ASCII code of special character that is used to control miniterm (menu), default 20 (DEC)

--quiet

Diagnostics: suppress non-error messages, default Off

-d, --project-dir

Specify the path to project directory. By default, --project-dir is equal to current working directory (CWD).

-e, --environment

Process specified environments.

You can also specify which environments should be processed by default using *env_default* option from *Project Configuration File platformio.ini*.

Examples

1. Show available options for *monitor*

```
$ platformio device monitor --help
Usage: platformio device monitor [OPTIONS]

Options:
  -p, --port TEXT      Port, a number or a device name
  -b, --baud INTEGER   Set baud rate, default=9600
  --parity [N|E|O|S|M] Set parity, default=N
  --rtscts              Enable RTS/CTS flow control, default=Off
  --xonxoff             Enable software flow control, default=Off
  --rts [0|1]            Set initial RTS line state, default=0
  --dtr [0|1]            Set initial DTR line state, default=0
  --echo                Enable local echo, default=Off
  --encoding TEXT       Set the encoding for the serial port (e.g. hexlify,
                        Latin1, UTF-8), default: UTF-8
  -f, --filter TEXT     Add text transformation
  --eol [CR|LF|CRLF]    End of line mode, default=CRLF
  --raw                 Do not apply any encodings/transformations
  --exit-char INTEGER   ASCII code of special character that is used to exit
                        the application, default=29 (DEC)
  --menu-char INTEGER   ASCII code of special character that is used to
                        control miniterm (menu), default=20 (DEC)
  --quiet               Diagnostics: suppress non-error messages, default=Off
  -h, --help             Show this message and exit.
```

2. Communicate with serial device and print help inside terminal

```
$ platformio device monitor

--- Available ports:
--- /dev/cu.Bluetooth-Incoming-Port n/a
--- /dev/cu.Bluetooth-Modem n/a
--- /dev/cu.SLAB_USBtoUART CP2102 USB to UART Bridge Controller
--- /dev/cu.obd2ecu-SPPDev n/a
Enter port name:/dev/cu.SLAB_USBtoUART
--- Miniterm on /dev/cu.SLAB_USBtoUART: 9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Hello PlatformIO!
---
--- Ctrl+]  Exit program
--- Ctrl+T  Menu escape key, followed by:
```

(continues on next page)

(continued from previous page)

```
---- Menu keys:
----   Ctrl+T Send the menu character itself to remote
----   Ctrl+] Send the exit character itself to remote
----   Ctrl+I Show info
----   Ctrl+U Upload file (prompt will be shown)
---- Toggles:
----   Ctrl+R RTS           Ctrl+E local echo
----   Ctrl+D DTR           Ctrl+B BREAK
----   Ctrl+L line feed     Ctrl+A Cycle repr mode
---- 

---- Port settings (Ctrl+T followed by the following):
----   p             change port
----   7 8         set data bits
----   n e o s m   change parity (None, Even, Odd, Space, Mark)
----   1 2 3       set stop bits (1, 2, 1.5)
----   b             change baud rate
----   x X           disable/enable software flow control
----   r R           disable/enable hardware flow control
---- exit ---
```

platformio home

Helper command for *PlatformIO Home*.

Contents

- *platformio home*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio home
pio home
```

Description

Launch *PlatformIO Home* Web-server.

Options

--port

Web-server HTTP port, default is 8008.

--host

Web-server HTTP host, default is 127.0.0.1. You can open PIO Home for inbound connections using host 0.0.0.0.

--no-open

Do not automatically open PIO Home in a system Web-browser.

Examples

```
> platformio home  
  
____I_____  
/ \_---\ PlatformIO Home  
/ \_\_--\ /  
| [] | [ ] | http://127.0.0.1:8008  
|__|_____|
```

platformio init

Contents

- *platformio init*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio init [OPTIONS]  
pio init [OPTIONS]
```

Description

Initialize new PlatformIO based project or update existing with new data.

This command will create:

- *Project Configuration File* `platformio.ini`
- `src` directory where you should place source code (`*.h`, `*.c`, `*.cpp`, `*.S`, `*.ino`, etc.)
- `lib` directory can be used for the project specific (private) libraries. More details are located in `lib/readme.txt` file.
- Miscellaneous files for VCS and *Continuous Integration* support.

Options

-d, --project-dir

A path to a directory where *PlatformIO* will initialize new project.

-b, --board

If you specify board ID (you can pass multiple `--board` options), then *PlatformIO* will automatically generate environment for *Project Configuration File* `platformio.ini` and pre-fill these data:

- *platform*
- *framework*
- *board*

The full list with pre-configured boards is available here *Development Platforms*.

--ide

Initialize PlatformIO project for the specified IDE which can be imported later via “Import Project” functionality.

A list with supported IDE is available within `platformio init --help` command. Also, please take a look at *Cloud & Desktop IDE* page.

-O, --project-option

Initialize project with additional options from *Project Configuration File* `platformio.ini`. For example, `platformio init --project-option="lib_deps=ArduinoJSON"`. Multiple options are allowed.

--env-prefix

An environment prefix which will be used with pair in board type. For example, the default environment name for `teensy_31` board will be `[env:teensy_31]`.

-s, --silent

Suppress progress reporting

Examples

1. Initialize new project in a current working directory

```
> platformio init

The current working directory *** will be used for the new project.
You can specify another project directory via
`platformio init -d %PATH_TO_THE_PROJECT_DIR%` command.

The next files/directories will be created in ***
platformio.ini - Project Configuration File. |-- PLEASE EDIT ME <-|
src - Put your source files here
lib - Put here project specific (private) libraries
Project has been successfully initialized!
Useful commands:
`platformio run` - process/build project from the current directory
`platformio run --target upload` or `platformio run -t upload` - upload firmware to
`embedded board
`platformio run --target clean` - clean project (remove compiled files)
```

2. Initialize new project in a specified directory

```
> platformio init -d %PATH_TO_DIR%  
  
The next files/directories will be created in ***  
platformio.ini - Project Configuration File. |-> PLEASE EDIT ME <-|  
...  
...
```

3. Initialize project for Arduino Uno

```
> platformio init --board uno  
  
The current working directory *** will be used for the new project.  
You can specify another project directory via  
`platformio init -d %PATH_TO_THE_PROJECT_DIR%` command.  
...  
...
```

4. Initialize project for Teensy 3.1 board with custom *mbed*

```
> platformio init --board teensy31 --project-option "framework=mbed"  
  
The current working directory *** will be used for the new project.  
You can specify another project directory via  
`platformio init -d %PATH_TO_THE_PROJECT_DIR%` command.  
...  
...
```

Library Manager

Usage

```
platformio lib [OPTIONS] COMMAND  
  
# To print all available commands and options use  
platformio lib --help  
platformio lib COMMAND --help
```

Options

-g, --global

New in version 3.0.

Manage global PlatformIO's library storage ("home_dir/lib") where *Library Dependency Finder (LDF)* will look for dependencies by default.

-d, --storage-dir

New in version 3.0.

Manage custom library storage. It can be used later for the *lib_extra_dirs* option from *Project Configuration File platformio.ini*.

Demo

Commands

platformio lib builtin

Contents

- *platformio lib builtin*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio lib builtin [OPTIONS]
pio lib builtin [OPTIONS]
```

Description

List built-in libraries based on installed *Development Platforms* and their frameworks, SDKs, etc.

Options

--storage

List libraries from specified storages. For example, `framework-arduinoavr`.

--json-output

Return the output in `JSON` format

Examples

```
> platformio lib builtin
framework-arduinoavr
*****
Bridge
=====
Enables the communication between the Linux processor and the microcontroller. For Arduino/Genuino Yún, Yún Shield and TRE only.

Version: 1.6.1
Homepage: http://www.arduino.cc/en/Reference/YunBridgeLibrary
Keywords: communication
```

(continues on next page)

(continued from previous page)

```
Compatible frameworks: arduino
Compatible platforms: *
Authors: Arduino

EEPROM
=====
Enables reading and writing to the permanent board storage.

Version: 2.0
Homepage: http://www.arduino.cc/en/Reference/EEPROM
Keywords: data, storage
Compatible frameworks: arduino
Compatible platforms: atmelavr
Authors: Arduino, Christopher Andrews

...

framework-arduinomas
*****
```

```
Audio
=====
Allows playing audio files from an SD card. For Arduino DUE only.

Version: 1.0
Homepage: http://arduino.cc/en/Reference/Audio
Keywords: signal, input, output
Compatible frameworks: arduino
Compatible platforms: atmelsam
Authors: Arduino

...

framework-arduinoespressif32
*****
```

```
SPI
===
Enables the communication with devices that use the Serial Peripheral Interface (SPI) ↳  
↳ Bus. For all Arduino boards, BUT Arduino DUE.

Version: 1.0
Homepage: http://arduino.cc/en/Reference/SPI
Keywords: signal, input, output
Compatible frameworks: arduino
Compatible platforms:
Authors: Hristo Gochkov

...

framework-arduinoespressif8266
*****
```

```
ArduinoOTA
=====
```

(continues on next page)

(continued from previous page)

Enables Over The Air upgrades, via wifi `and` `espota.py` UDP request/TCP download.

Version: 1.0

Keywords: communication

Compatible frameworks: arduino

Compatible platforms: espressif8266

Authors: Ivan Grokhotkov `and` Miguel Angel Ajo

DNSServer

=====

A simple DNS server `for` ESP8266.

Version: 1.1.0

Keywords: communication

Compatible frameworks: arduino

Compatible platforms: espressif8266

Authors: Kristijan Novoselić

...

framework-arduinointel

Adafruit NeoPixel

=====

Arduino library `for` controlling single-wire-based LED pixels `and` strip.

Version: 1.0.3

Homepage: https://github.com/adafruit/Adafruit_NeoPixel

Keywords: display

Compatible frameworks: arduino

Compatible platforms: *

Authors: Adafruit

CurieBLE

=====

Library to manage the Bluetooth Low Energy module `with` Curie Core boards.

Version: 1.0

Keywords: communication

Compatible frameworks: arduino

Compatible platforms: intel_arc32

Authors: Emutex

CurieEEPROM

=====

Enables reading `and` writing to OTP flash area of Curie

Version: 1.0

Homepage: <http://www.arduino.cc/en/Reference/EEPROM>

Keywords: data, storage

Compatible frameworks: arduino

Compatible platforms: intel_arc32

Authors: Intel

...

(continues on next page)

(continued from previous page)

```
framework-arduinomicrochippic32
*****
Firmata
=====
Enables the communication with computer apps using a standard serial protocol. For all Arduino boards.

Version: 2.4.4
Homepage: https://github.com/firmata/arduino
Keywords: device, control
Compatible frameworks: arduino
Compatible platforms: *
Authors: Firmata Developers

framework-arduinoteensy
*****
Adafruit CC3000 Library
=====
Library code for Adafruit's CC3000 WiFi breakouts.

Version: 1.0.1
Homepage: https://github.com/adafruit/Adafruit\_CC3000\_Library
Keywords: communication
Compatible frameworks: arduino
Compatible platforms: *
Authors: Adafruit

...
framework-energiamsp430
*****
AIR430BoostEuropeETSI
=====
Library for the CC110L Sub-1GHz radio BoosterPack for use in Europe

Version: 1.0.0
Homepage: http://energia.nu/reference/libraries/
Keywords: communication
Compatible frameworks: arduino
Compatible platforms:
Authors: Energia

...
framework-energiativa
*****
aJson
=====
An Arduino library to enable JSON processing with Arduino

Keywords: json, rest, http, web
Compatible frameworks: arduino
Compatible platforms: atmelavr
```

platformio lib install

Contents

- *platformio lib install*
 - *Usage*
 - *Description*
 - *Storage Options*
 - *Options*
 - *Version control*
 - * *Git*
 - * *Mercurial*
 - * *Subversion*
 - *Examples*

Usage

```
platformio lib [STORAGE_OPTIONS] install [OPTIONS] [LIBRARY...]
pio lib [STORAGE_OPTIONS] install [OPTIONS] [LIBRARY...]

# install project dependent library
# (run it from a project root where is located "platformio.ini")
platformio lib install [OPTIONS] [LIBRARY...]

# install to global storage
platformio lib --global install [OPTIONS] [LIBRARY...]
platformio lib -g install [OPTIONS] [LIBRARY...]

# install to custom storage
platformio lib --storage-dir /path/to/dir install [OPTIONS] [LIBRARY...]
platformio lib -d /path/to/dir install [OPTIONS] [LIBRARY...]

# [LIBRARY...] forms
platformio lib [STORAGE_OPTIONS] install (with no args, project dependencies)
platformio lib [STORAGE_OPTIONS] install <id>
platformio lib [STORAGE_OPTIONS] install id=<id>
platformio lib [STORAGE_OPTIONS] install <id>@<version>
platformio lib [STORAGE_OPTIONS] install <id>@<version range>
platformio lib [STORAGE_OPTIONS] install <name>
platformio lib [STORAGE_OPTIONS] install <name>@<version>
platformio lib [STORAGE_OPTIONS] install <name>@<version range>
platformio lib [STORAGE_OPTIONS] install <zip or tarball url>
platformio lib [STORAGE_OPTIONS] install file://<zip or tarball file>
platformio lib [STORAGE_OPTIONS] install file://<folder>
platformio lib [STORAGE_OPTIONS] install <repository>
platformio lib [STORAGE_OPTIONS] install <name>=<repository> (name it should have
˓→locally)
platformio lib [STORAGE_OPTIONS] install <repository#tag> ("tag" can be commit,
˓→branch or tag)
```

(continues on next page)

(continued from previous page)

Warning: If some libraries are not visible in *PlatformIO IDE* and Code Completion or Code Linting does not work properly, please perform

- **Atom:** “Menu: PlatformIO > Rebuild C/C++ Project Index (Autocomplete, Linter)”
- **VSCODE:** “Menu: View > Command Palette... > PlatformIO: Rebuild C/C++ Project Index”

Description

Install a library, and any libraries that it depends on using:

1. Library id or name from [PlatformIO Library Registry](#)
2. Custom folder, repository or archive.

The version supports Semantic Versioning (<major>.<minor>.<patch>) and can take any of the following forms:

- 0.1.2 - an exact version number. Use only this exact version
- ^0.1.2 - any compatible version (exact version for 0.x.x versions)
- ~0.1.2 - any version with the same major and minor versions, and an equal or greater patch version
- >0.1.2 - any version greater than 0.1.2. >=, <, and <= are also possible
- >0.1.0, !=0.2.0, <0.3.0 - any version greater than 0.1.0, not equal to 0.2.0 and less than 0.3.0

PlatformIO supports installing from local directory or archive. Need to use `file://` prefix before local path. Also, directory or archive should contain `.library.json` manifest (see [library.json](#)).

- `file:///local/path/to/the/platform/dir`
- `file:///local/path/to/the/platform.zip`
- `file:///local/path/to/the/platform.tar.gz`

Storage Options

See base options for [Library Manager](#).

Options

-s, --silent

Suppress progress reporting

--interactive

Allow to make a choice for all prompts

-f, --force

Reinstall/redownload library if it exists

Version control

PlatformIO supports installing from Git, Mercurial and Subversion, and detects the type of VCS using url prefixes: “git+”, “hg+”, or “svn+”.

Note: PlatformIO requires a working VCS command on your path: git, hg or svn.

Git

The supported schemes are: git, git+https and git+ssh. Here are the supported forms:

- user/library (short version for GitHub repository)
- <https://github.com/user/library.git>
- git+git://git.server.org/my-library
- git+https://git.server.org/my-library
- git+ssh://git.server.org/my-library
- git+ssh://user@git.server.org/my-library
- [user@]host.xz:path/to/repo.git

Passing branch names, a commit hash or a tag name is possible like so:

- <https://github.com/user/library.git#master>
- git+git://git.server.org/my-library#master
- git+https://git.server.org/my-library#v1.0
- git+ssh://git.server.org/my-library#7846d8ad52f983f2f2887bdc0f073fe9755a806d

Mercurial

The supported schemes are: hg+http, hg+https and hg+ssh. Here are the supported forms:

- <https://developer.mbed.org/users/user/code/library/> (install ARM mbed library)
- hg+hg://hg.server.org/my-library
- hg+https://hg.server.org/my-library
- hg+ssh://hg.server.org/my-library

Passing branch names, a commit hash or a tag name is possible like so:

- hg+hg://hg.server.org/my-library#master
- hg+https://hg.server.org/my-library#v1.0
- hg+ssh://hg.server.org/my-library#4cfe2fa00668

Subversion

The supported schemes are: svn, svn+svn, svn+http, svn+https and svn+ssh. Here are the supported forms:

- svn+svn://svn.server.org/my-library
- svn+https://svn.server.org/my-library
- svn+ssh://svn.server.org/my-library

You can also give specific revisions to an SVN URL, like so:

- svn+svn://svn.server.org/my-library#13

Examples

1. Install the latest version of library to a global storage using ID or NAME

```
> platformio lib -g install 4

Library Storage: /storage/dir/...
LibraryManager: Installing id=4
Downloading [########################################] 100%
Unpacking [########################################] 100%
IRremote @ 2.2.1 has been successfully installed!

# repeat command with name
> platformio lib -g install IRRemote

Library Storage: /storage/dir/...
Looking for IRRemote library in registry
Found: https://platformio.org/lib/show/4/IRremote
LibraryManager: Installing id=4
IRremote @ 2.2.1 is already installed
```

2. Install specified version of a library to a global storage

```
> platformio lib -g install ArduinoJson@5.6.7

Library Storage: /storage/dir/...
Looking for ArduinoJson library in registry
Found: https://platformio.org/lib/show/64/ArduinoJson
LibraryManager: Installing id=64 @ 5.6.7
Downloading [########################################] 100%
Unpacking [########################################] 100%
ArduinoJson @ 5.6.7 has been successfully installed!
```

3. Install library with dependencies to custom storage

```
> platformio lib --storage-dir /my/storage/dir install DallasTemperature

Library Storage: /my/storage/dir
Looking for DallasTemperature library in registry
Found: https://platformio.org/lib/show/54/DallasTemperature
LibraryManager: Installing id=54
Downloading [########################################] 100%
Unpacking [########################################] 100%
```

(continues on next page)

(continued from previous page)

```
DallasTemperature @ 3.7.7 has been successfully installed!
Installing dependencies
Looking for OneWire library in registry
Found: https://platformio.org/lib/show/1/OneWire
LibraryManager: Installing id=1
Downloading [########################################] 100%
Unpacking [########################################] 100%
OneWire @ 8fd2ebfec7 has been successfully installed!
```

4. Install ARM mbed library to the global storage

```
> platformio lib -g install https://developer.mbed.org/users/simon/code/TextLCD/
Library Storage: /storage/dir/...
LibraryManager: Installing TextLCD
Mercurial Distributed SCM (version 3.8.4)
(see https://mercurial-scm.org for more information)

Copyright (C) 2005-2016 Matt Mackall and others
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
requesting all changes
adding changesets
adding manifests
adding file changes
added 9 changesets with 18 changes to 6 files
updating to branch default
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
TextLCD @ 308d188a2d3a has been successfully installed!
```

5. Install from archive using URL

```
> platformio lib -g install https://github.com/adafruit/DHT-sensor-library/archive/
master.zip

Library Storage: /storage/dir/...
LibraryManager: Installing master
Downloading [########################################] 100%
Unpacking [########################################] 100%
DHT sensor library @ 1.2.3 has been successfully installed!
```

platformio lib list

Contents

- *platformio lib list*
 - *Usage*
 - *Description*
 - *Storage Options*
 - *Options*

– Examples

Usage

```
platformio lib [STORAGE_OPTIONS] list [OPTIONS]
pio lib [STORAGE_OPTIONS] list [OPTIONS]

# list project dependent libraries
# (run it from a project root where is located "platformio.ini")
platformio lib list [OPTIONS]

# list libraries from global storage
platformio lib --global list [OPTIONS]
platformio lib -g list [OPTIONS]

# list libraries from custom storage
platformio lib --storage-dir /path/to/dir list [OPTIONS]
platformio lib -d /path/to/dir list [OPTIONS]
```

Description

List installed libraries

Storage Options

See base options for *Library Manager*.

Options

--json-output

Return the output in **JSON** format

Examples

```
> platformio lib -g list

Library Storage: /storage/dir/...
=====
Adafruit Unified Sensor
=====
#ID: 31
Required for all Adafruit Unified Sensor based libraries.

Version: 1.0.2
Keywords: sensors
Compatible frameworks: arduino
Compatible platforms: atmelavr, atmelsam, espressif8266, intel_arc32, microchippic32,
↳ nordicnrf51, teensy, timsp430
```

(continues on next page)

(continued from previous page)

```

Authors: Adafruit

ArduinoJson
=====
# ID: 64
An elegant and efficient JSON library for embedded systems

Version: 5.8.0
Keywords: web, json, http, rest
Compatible frameworks: arduino
Compatible platforms: atmelavr, atmelsam, espressif8266, intel_arc32, microchippic32, ↵nordicnrf51, teensy, timsp430
Authors: Benoit Blanchon

ArduinoJson
=====
# ID: 64
An elegant and efficient JSON library for embedded systems

Version: 5.6.7
Keywords: web, json, http, rest
Compatible frameworks: arduino
Compatible platforms: atmelavr, atmelsam, espressif8266, intel_arc32, microchippic32, ↵nordicnrf51, teensy, timsp430
Authors: Benoit Blanchon

ArduinoJson
=====
# ID: 64
An elegant and efficient JSON library for embedded systems

Version: 5.7.2
Keywords: web, json, http, rest
Compatible frameworks: arduino
Compatible platforms: atmelavr, atmelsam, espressif8266, intel_arc32, microchippic32, ↵nordicnrf51, teensy, timsp430
Authors: Benoit Blanchon

Blynk
=====
# ID: 415
Build a smartphone app for your project in minutes. Blynk allows creating IoT, ↵solutions easily. It supports WiFi, BLE, Bluetooth, Ethernet, GSM, USB, Serial. ↵Works with many boards like ESP8266, ESP32, Arduino UNO, Nano, Due, Mega, Zero, ↵MKR100, Yun, Raspberry Pi, Particle, Energia, ARM mbed, Intel Edison/Galileo/Joule, ↵BBC micro:bit, DFRobot, RedBearLab, Microduino, LinkIt ONE ...

Version: 0.4.3
Homepage: http://blynk.cc
Keywords: control, gprs, protocol, communication, app, bluetooth, serial, cloud, web, ↵usb, m2m, ble, 3g, smartphone, http, iot, device, sensors, data, esp8266, mobile, ↵wifi, ethernet, gsm
Compatible frameworks: energia, wiringpi, arduino
Compatible platforms: atmelavr, atmelsam, espressif8266, intel_arc32, linux_arm, ↵microchippic32, nordicnrf51, teensy, timsp430, titiva
Authors: Volodymyr Shymanskyy

```

(continues on next page)

(continued from previous page)

```
Bounce2
=====
#ID: 1106
Debouncing library for Arduino or Wiring

Version: 2.1
Keywords: input, signal, ouput, bounce
Compatible frameworks: arduino
Compatible platforms: atmelavr, atmelsam, espressif8266, intel_arc32, microchippic32, nordicnrf51, teensy, timsp430
Authors: Thomas O Fredericks

Homie
=====
#ID: 555
ESP8266 framework for Homie, a lightweight MQTT convention for the IoT

Version: 1.5.0
Keywords: home, mqtt, iot, esp8266, automation
Compatible frameworks: arduino
Compatible platforms: espressif8266
Authors: Marvin Roger

JustWifi
=====
#ID: 1282
Wifi Manager for ESP8266 that supports multiple wifi networks and scan for strongest signal

Version: 1.1.1
License: GPL-3.0
Keywords: manager, wifi, scan
Compatible frameworks: arduino
Compatible platforms: espressif8266
Authors: Xose Perez

LiquidCrystal
=====
#ID: 136
LiquidCrystal Library is faster and extensable, compatible with the original LiquidCrystal library

Version: 1.3.4
Keywords: lcd, hd44780
Compatible frameworks: arduino
Compatible platforms: atmelavr
Authors: F Malpartida

TextLCD
=====
hg+https://developer.mbed.org/users/simon/code/TextLCD/

Version: 308d188a2d3a
Keywords: uncategorized

Time
=====
```

(continues on next page)

(continued from previous page)

```
#ID: 44
Time keeping library

Version: 1.5
Homepage: http://playground.arduino.cc/Code/Time
Keywords: week, rtc, hour, year, month, second, time, date, day, minute
Compatible frameworks: arduino
Compatible platforms:
Authors: Michael Margolis, Paul Stoffregen

Timezone
=====
#ID: 76
Arduino library to facilitate time zone conversions and automatic daylight saving↔(summer) time adjustments

Version: 510ae2f6b6
Keywords: zone, time
Compatible frameworks: arduino
Compatible platforms: atmelavr
Authors: Jack Christensen

U8g2
=====
#ID: 942
Monochrome LCD, OLED and eInk Library. Display controller: SSD1305, SSD1306, SSD1322,↔SSD1325, SSD1327, SSD1606, SH1106, T6963, RA8835, LC7981, PCD8544, PCF8812, UC1604,↔UC1608, UC1610, UC1611, UC1701, ST7565, ST7567, NT7534, ST7920, LD7032, KS0108.↔Interfaces: I2C, SPI, Parallel.

Version: 2.11.4
Homepage: https://github.com/olikraus/u8g2
Keywords: display
Compatible frameworks: arduino
Compatible platforms: atmelavr, atmelsam, espressif8266, intel_arc32, microchippic32,↔nordicnrf51, teensy, timsp430
Authors: oliver

USB-Host-Shield-20
=====
#ID: 59
Revision 2.0 of MAX3421E-based USB Host Shield Library

Version: 1.2.1
License: GPL-2.0
Keywords: usb, spp, mass storage, pl2303, acm, ftdi, xbox, host, hid, wii, buzz, ps3,↔bluetooth, adk, ps4
Compatible frameworks: spl, arduino
Compatible platforms: atmelavr, atmelsam, teensy, nordicnrf51, ststm32
Authors: Oleg Mazurov, Alexei Glushchenko, Kristian Lauszus, Andrew Kroll
```

platformio lib register

Contents

- *platformio lib register*
 - *Usage*
 - *Description*
 - *Examples*

Usage

```
platformio lib register [MANIFEST_URL]
pio lib register [MANIFEST_URL]
```

Description

Register new library in PlatformIO Library Registry.

PlatformIO Library Registry supports the next library manifests:

- PlatformIO *library.json*
- Arduino *library.properties*
- ARM mbed yotta *module.json*.

Examples

```
platformio lib register https://raw.githubusercontent.com/bblanchon/ArduinoJson/
˓→master/library.json
platformio lib register https://raw.githubusercontent.com/adafruit/DHT-sensor-library/
˓→master/library.properties
platformio lib register https://raw.githubusercontent.com/ARMmbed/ble/master/module.
˓→json
```

platformio lib search

Contents

- *platformio lib search*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio lib search [OPTIONS] [QUERY]
pio lib search [OPTIONS] [QUERY]
```

Description

Search for library in [PlatformIO Library Registry](#) by *library.json* fields in the boolean mode.

The boolean search capability supports the following operators:

Operator	Description
+	A leading or trailing plus sign indicates that this word must be present in library fields (see above) that is returned.
-	A leading or trailing minus sign indicates that this word must not be present in any of the libraries that are returned.
(no operator)	By default (when neither + nor - is specified), the word is optional, but the libraries that contain it are rated higher.
> <	These two operators are used to change a word's contribution to the relevance value that is assigned to a library. The > operator increases the contribution and the < operator decreases it.
()	Parentheses group words into subexpressions. Parenthesized groups can be nested.
~	A leading tilde acts as a negation operator, causing the word's contribution to the library's relevance to be negative. This is useful for marking "noise" words. A library containing such a word is rated lower than others, but is not excluded altogether, as it would be with the - operator.
*	The asterisk serves as the truncation (or wildcard) operator. Unlike the other operators, it is appended to the word to be affected. Words match if they begin with the word preceding the * operator.
"	A phrase that is enclosed within double quote ("") characters matches only libraries that contain the phrase literally, as it was typed.

For more detail information please go to [MySQL Boolean Full-Text Searches](#).

Options

--id

Filter libraries by registry ID

-n, --name

Filter libraries by specified name (strict search)

-a, --author

Filter libraries by specified author

-k, --keyword

Filter libraries by specified keyword

-f, --framework

Filter libraries by specified framework

-p, --platform

Filter libraries by specified keyword

-i, --header

Filter libraries by header file (include)

For example, platformio lib search --header "OneWire.h"

--json-output

Return the output in **JSON** format

--page

Manually paginate through search results. This option is useful in pair with --json-output.

Examples

1. List all libraries

```
> platformio lib search

Found N libraries:

ArduinoJson
=====
#ID: 64
An elegant and efficient JSON library for embedded systems

Keywords: web, json, http, rest
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Microchip↪PIC32, Nordic nRF51, Teensy, TI MSP430
Authors: Benoit Blanchon

DHT sensor library
=====
#ID: 19
Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors

Keywords: unified, dht, sensor, temperature, humidity
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR
Authors: Adafruit Industries

PubSubClient
=====
#ID: 89
A client library for MQTT messaging. MQTT is a lightweight messaging protocol ideal↪for small devices. This library allows you to send and receive MQTT messages. It↪supports the latest MQTT 3.1.1 protocol and can be configured to use the older MQTT↪3.1...

Keywords: ethernet, mqtt, iot, m2m
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Microchip↪PIC32, Nordic nRF51, Teensy, TI MSP430
Authors: Nick O'Leary
```

(continues on next page)

(continued from previous page)

```
...
ESPAsyncWebServer
=====
#ID: 306
Asynchronous HTTP and WebSocket Server Library for ESP8266 and ESP32

Keywords: async, websocket, http, webserver
Compatible frameworks: Arduino
Compatible platforms: Espressif 8266
Authors: Hristo Gochkov

Show next libraries? [y/N]:
...
```

2. Search for 1-Wire libraries

```
> platformio lib search "1-wire"

Found N libraries:

DS1820
=====
#ID: 196
Dallas / Maxim DS1820 1-Wire library. For communication with multiple DS1820 on a
single 1-Wire bus. Also supports DS18S20 and DS18B20.

Keywords: ds18s20, 1-wire, ds1820, ds18b20
Compatible frameworks: mbed
Compatible platforms: Freescale Kinetis, Nordic nRF51, NXP LPC, ST STM32, Teensy
Authors: Michael Hagberg

OneWire
=====
#ID: 1
Control 1-Wire protocol (DS18S20, DS18B20, DS2408 and etc)

Keywords: onewire, temperature, bus, 1-wire, ibutton, sensor
Compatible frameworks: Arduino
Compatible platforms:
Authors: Paul Stoffregen, Jim Studt, Tom Pollard, Derek Yerger, Josh Larios, Robin
James, Glenn Trewitt, Jason Dangel, Guillermo Lovato, Ken Butcher, Mark Tillotson,
Bertrik Sikken, Scott Roberts

Show next libraries? [y/N]:
...
```

3. Search for Arduino-based “I2C” libraries

```
> platformio lib search "i2c" --framework="arduino"

Found N libraries:

I2Cdevlib-AK8975
=====
#ID: 10
AK8975 is 3-axis electronic compass IC with high sensitive Hall sensor technology
(continues on next page)
```

(continued from previous page)

```

Keywords: i2c, i2cdevlib, sensor, compass
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR
Authors: Jeff Rowberg

I2Cdevlib-Core
=====
#ID: 11
The I2C Device Library (I2Cdevlib) is a collection of uniform and well-documented_
→classes to provide simple and intuitive interfaces to I2C devices.

Keywords: i2cdevlib, i2c
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR
Authors: Jeff Rowberg

Adafruit 9DOF Library
=====
#ID: 14
Unified sensor driver for the Adafruit 9DOF Breakout (L3GD20 / LSM303)

Keywords: magnetometer, unified, accelerometer, spi, compass, i2c, sensor, gyroscope
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR
Authors: Adafruit Industries

Show next libraries? [y/N]:
...

```

4. Search for libraries by “web” and “http” keywords.

```

> platformio lib search --keyword="web" --keyword="http"

Found N libraries:

ArduinoJson
=====
#ID: 64
An elegant and efficient JSON library for embedded systems

Keywords: web, json, http, rest
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Microchip_
→PIC32, Nordic nRF51, Teensy, TI MSP430
Authors: Benoit Blanchon

ESPAsyncWebServer
=====
#ID: 306
Asynchronous HTTP and WebSocket Server Library for ESP8266 and ESP32

Keywords: async, websocket, http, webserver
Compatible frameworks: Arduino
Compatible platforms: Espressif 8266
Authors: Hristo Gochkov

```

(continues on next page)

(continued from previous page)

```

ESP8266wifi
=====
#ID: 1101
ESP8266 Arduino library with built in reconnect functionality

Keywords: web, http, wifi, server, client, wi-fi
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR
Authors: Jonas Ekstrand

Blynk
=====
#ID: 415
Build a smartphone app for your project in minutes. Blynk allows creating IoT
→solutions easily. It supports WiFi, BLE, Bluetooth, Ethernet, GSM, USB, Serial.
→Works with many boards like ESP8266, ESP32, Arduino UNO, Nano, Due, Mega, Zero,
→MKR100, Yun, ...

Keywords: control, gprs, protocol, communication, app, bluetooth, serial, cloud, web,
→usb, m2m, ble, 3g, smartphone, http, iot, device, sensors, data, esp8266, mobile,
→wifi, ethernet, gsm
Compatible frameworks: Arduino, Energia, WiringPi
Compatible platforms: Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Linux ARM,
→Microchip PIC32, Nordic nRF51, Teensy, TI MSP430, TI Tiva
Authors: Volodymyr Shymanskyy

Show next libraries? [y/N]:
...

```

5. Search for libraries by “Adafruit Industries” author

```

> platformio lib search --author="Adafruit Industries"

Found N libraries:

DHT sensor library
=====
#ID: 19
Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors

Keywords: unified, dht, sensor, temperature, humidity
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR
Authors: Adafruit Industries

Adafruit DHT Unified
=====
#ID: 18
Unified sensor library for DHT (DHT11, DHT22 and etc) temperature and humidity sensors

Keywords: unified, dht, sensor, temperature, humidity
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR
Authors: Adafruit Industries

Show next libraries? [y/N]:
...

```

6. Search for libraries which are compatible with Dallas temperature sensors like DS18B20, DS18S20 and etc.

```
> platformio lib search "DS*"

Found N libraries:

DS1820
=====
#ID: 196
Dallas / Maxim DS1820 1-Wire library. For communication with multiple DS1820 on a
single 1-Wire bus. Also supports DS18S20 and DS18B20.

Keywords: ds18s20, 1-wire, ds1820, ds18b20
Compatible frameworks: mbed
Compatible platforms: Freescale Kinetis, Nordic nRF51, NXP LPC, ST STM32, Teensy
Authors: Michael Hagberg

I2Cdevlib-DS1307
=====
#ID: 99
The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal
(BCD) clock/calendar plus 56 bytes of NV SRAM

Keywords: i2cdevlib, clock, i2c, rtc, time
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR
Authors: Jeff Rowberg

Show next libraries? [y/N]:
...
```

7. Search for [Energia-based *nRF24*](#) or [*HttpClient*](#) libraries. The search query that is described below can be interpreted like `energia nRF24 OR energia HttpClient`

```
> platformio lib search "+(nRF24 HttpClient)" --framework="energia"

Found N libraries:

nRF24
=====
#ID: 43
The nRF24L01 is a low-cost 2.4GHz ISM transceiver module. It supports a number of
channel frequencies in the 2.4GHz band and a range of data rates.

Keywords: wireless, spi, rf, radio
Compatible frameworks: Energia
Compatible platforms: TI MSP430
Authors: Eric

HttpClient
=====
#ID: 46
HttpClient is a library to make it easier to interact with web servers

Keywords: web, client, http, ethernet
Compatible frameworks: Energia
Compatible platforms: TI MSP430, TI Tiva
Authors: Zack Lalanne
```

(continues on next page)

(continued from previous page)

```

RadioHead
=====
#ID: 124
The RadioHead Packet Radio library which provides a complete object-oriented library for sending and receiving packetized messages via RF22/24/26/27/69, Si4460/4461/4463/4464, nRF24/nRF905, SX1276/77/78, RFM95/96/97/98 and etc.

Keywords: wireless, rf, radio
Compatible frameworks: Arduino, Energia
Compatible platforms: Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Microchip PIC32, Nordic nRF51, Teensy, TI MSP430, TI Tiva
Authors: Mike McCauley

Show next libraries? [y/N]:
...

```

8. Search for the all sensor libraries excluding temperature.

```

> platformio lib search "sensor -temperature"

Found N libraries:

SparkFun VL6180 Sensor
=====
#ID: 407
The VL6180 combines an IR emitter, a range sensor, and an ambient light sensor together for you to easily use and communicate with via an I2C interface.

Keywords: sensors
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Microchip PIC32, Nordic nRF51, Teensy, TI MSP430
Authors: Casey Kuhns@SparkFun, SparkFun Electronics

I2Cdevlib-AK8975
=====
#ID: 10
AK8975 is 3-axis electronic compass IC with high sensitive Hall sensor technology

Keywords: i2c, i2cdevlib, sensor, compass
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR
Authors: Jeff Rowberg

Adafruit 9DOF Library
=====
#ID: 14
Unified sensor driver for the Adafruit 9DOF Breakout (L3GD20 / LSM303)

Keywords: magnetometer, unified, accelerometer, spi, compass, i2c, sensor, gyroscope
Compatible frameworks: Arduino
Compatible platforms: Atmel AVR
Authors: Adafruit Industries

Show next libraries? [y/N]:
...

```

platformio lib show

Contents

- *platformio lib show*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio lib show [LIBRARY]
pio lib show [LIBRARY]
```

Description

Show detailed info about a library using PlatformIO Library Registry.

The possible values for [LIBRARY]:

- Library ID from Registry (preferred)
- Library Name

Options

--json-output

Return the output in JSON format

Examples

```
> platformio lib show OneWire

PubSubClient
=====
#ID: 89
A client library for MQTT messaging. MQTT is a lightweight messaging protocol ideal for small devices. This library allows you to send and receive MQTT messages. It supports the latest MQTT 3.1.1 protocol and can be configured to use the older MQTT 3.1...

Version: 2.6, released 10 months ago
Manifest: https://raw.githubusercontent.com/ivankravets/pubsubclient/patch-2/library.json
Homepage: http://pubsubclient.knolleary.net
```

(continues on next page)

(continued from previous page)

Repository: <https://github.com/knolleary/pubsubclient.git>

Authors

Nick O'Leary <https://github.com/knolleary>

Keywords

ethernet
mqtt
iot
m2m

Compatible frameworks

Arduino

Compatible platforms

Atmel AVR
Atmel SAM
Espressif **8266**
Intel ARC32
Microchip PIC32
Nordic nRF51
Teensy
TI MSP430

Headers

PubSubClient.h

Examples

http://dl.platformio.org/libraries/examples/0/89/mqtt_auth.ino
http://dl.platformio.org/libraries/examples/0/89/mqtt_basic.ino
http://dl.platformio.org/libraries/examples/0/89/mqtt_esp8266.ino
http://dl.platformio.org/libraries/examples/0/89/mqtt_publish_in_callback.ino
http://dl.platformio.org/libraries/examples/0/89/mqtt_reconnect_nonblocking.ino
http://dl.platformio.org/libraries/examples/0/89/mqtt_stream.ino

Versions

2.6, released **10** months ago

Downloads

Today: **25**
Week: **120**
Month: **462**

platformio lib stats

Contents

- *platformio lib stats*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio lib stats
pio lib stats
```

Description

Show PlatformIO Library Registry statistics:

- Recently updated
- Recently added
- Recent keywords
- Popular keywords
- Featured: Today
- Featured: Week
- Featured: Month

This information is the same that is shown on this page:

- <https://platformio.org/lib>

Options

--json-output

Return the output in **JSON** format

Examples

RECENTLY UPDATED		

Name	Date	Url
GroveEncoder	12 hours ago	https://platformio.org/lib/show/1382/GroveEncoder
GroveEncoder		

(continues on next page)

(continued from previous page)

RF24G	12 hours ago	https://platformio.org/lib/show/1381/
↳ RF24G		
Sim800L Library Revised	12 hours ago	https://platformio.org/lib/show/1380/
↳ Sim800L%20Library%20Revised		
ArduinoSTL	12 hours ago	https://platformio.org/lib/show/750/
↳ ArduinoSTL		
hd44780	13 hours ago	https://platformio.org/lib/show/738/
↳ hd44780		
RECENTLY ADDED		

Name	Date	Url
↳		
↳		
GroveEncoder	12 hours ago	https://platformio.org/lib/show/1382/
↳ GroveEncoder		
RF24G	12 hours ago	https://platformio.org/lib/show/1381/
↳ RF24G		
Sim800L Library Revised	12 hours ago	https://platformio.org/lib/show/1380/
↳ Sim800L%20Library%20Revised		
DS3231	a day ago	https://platformio.org/lib/show/1379/
↳ DS3231		
ArduboyPlaytune	4 days ago	https://platformio.org/lib/show/1378/
↳ ArduboyPlaytune		
RECENT KEYWORDS		

Name	Url	
↳		
↳		
cobs	https://platformio.org/lib/search?query=keyword%3Acobs	
packet	https://platformio.org/lib/search?query=keyword%3Apacket	
framing	https://platformio.org/lib/search?query=keyword%3Aframing	
3g	https://platformio.org/lib/search?query=keyword%3A3g	
tdd	https://platformio.org/lib/search?query=keyword%3Atdd	
POPULAR KEYWORDS		

Name	Url	
↳		
↳		
display	https://platformio.org/lib/search?query=keyword%3Adisplay	
lcd	https://platformio.org/lib/search?query=keyword%3Alcd	
sensors	https://platformio.org/lib/search?query=keyword%3Asensors	
graphics	https://platformio.org/lib/search?query=keyword%3Agraphics	
communication	https://platformio.org/lib/search?query=keyword%3Acommunication	
↳ %3Acommunication		
oled	https://platformio.org/lib/search?query=keyword%3Aoled	
tft	https://platformio.org/lib/search?query=keyword%3Atft	
control	https://platformio.org/lib/search?query=keyword%3Acontrol	
device	https://platformio.org/lib/search?query=keyword%3Adevice	
glcd	https://platformio.org/lib/search?query=keyword%3Aglcd	
displaycore	https://platformio.org/lib/search?query=keyword%3Adisplaycore	
font	https://platformio.org/lib/search?query=keyword%3Afont	

(continues on next page)

(continued from previous page)

other	https://platformio.org/lib/search?query=keyword%3Aother
i2c	https://platformio.org/lib/search?query=keyword%3Ai2c
input	https://platformio.org/lib/search?query=keyword%3Ainput
signal	https://platformio.org/lib/search?query=keyword%3Asignal
sensor	https://platformio.org/lib/search?query=keyword%3Asensor
output	https://platformio.org/lib/search?query=keyword%3Aoutput
spi	https://platformio.org/lib/search?query=keyword%3Aspi
data	https://platformio.org/lib/search?query=keyword%3Adata
timing	https://platformio.org/lib/search?query=keyword%3Atiming
serial	https://platformio.org/lib/search?query=keyword%3Aserial
temperature	https://platformio.org/lib/search?query=keyword%3Atemperature
http	https://platformio.org/lib/search?query=keyword%3Ahttp
wifi	https://platformio.org/lib/search?query=keyword%3Awifi
rf	https://platformio.org/lib/search?query=keyword%3Arf
i2cdevlib	https://platformio.org/lib/search?query=keyword%3Ai2cdevlib
processing	https://platformio.org/lib/search?query=keyword%3Aprocessing
storage	https://platformio.org/lib/search?query=keyword%3Astorage
radio	https://platformio.org/lib/search?query=keyword%3Aradio
web	https://platformio.org/lib/search?query=keyword%3Aweb
accelerometer	https://platformio.org/lib/search?query=keyword%3Aaccelerometer
↳ %3Aaccelerometer	
wireless	https://platformio.org/lib/search?query=keyword%3Awireless
protocol	https://platformio.org/lib/search?query=keyword%3Aprotocol
server	https://platformio.org/lib/search?query=keyword%3Aserver
wi-fi	https://platformio.org/lib/search?query=keyword%3Awi-fi
ethernet	https://platformio.org/lib/search?query=keyword%3Aethernet
mbed	https://platformio.org/lib/search?query=keyword%3Ambed
openag	https://platformio.org/lib/search?query=keyword%3Aopenag
led	https://platformio.org/lib/search?query=keyword%3Aled
esp8266	https://platformio.org/lib/search?query=keyword%3Aesp8266
humidity	https://platformio.org/lib/search?query=keyword%3Ahumidity
time	https://platformio.org/lib/search?query=keyword%3Atime
iot	https://platformio.org/lib/search?query=keyword%3Aiot
json	https://platformio.org/lib/search?query=keyword%3Ajson
timer	https://platformio.org/lib/search?query=keyword%3Atimer
client	https://platformio.org/lib/search?query=keyword%3Aclient
driver	https://platformio.org/lib/search?query=keyword%3Adriver
button	https://platformio.org/lib/search?query=keyword%3Abutton
mbed-official	https://platformio.org/lib/search?query=keyword%3Ambed-official
↳ official	
FEATURED: TODAY	

Name	Url
-----	-----
↳ -----	-----
↳ -----	-----
PubSubClient	https://platformio.org/lib/show/89/PubSubClient
Adafruit Unified Sensor	https://platformio.org/lib/show/31/Adafruit%20Unified%20Sensor
DHT sensor library	https://platformio.org/lib/show/19/DHT%20sensor%20library
ESPAsyncUDP	https://platformio.org/lib/show/359/ESPAsyncUDP
NtpClientLib	https://platformio.org/lib/show/727/NtpClientLib
Embedis	https://platformio.org/lib/show/408/Embedis
Blynk	https://platformio.org/lib/show/415/Blynk
SimpleTimer	https://platformio.org/lib/show/419/SimpleTimer
Adafruit DHT Unified	https://platformio.org/lib/show/18/Adafruit%20DHT%20Unified

(continues on next page)

(continued from previous page)

RTCLib	https://platformio.org/lib/show/83/RTCLib
FEATURED: WEEK	

Name	Url
-----	-----
↳	-----
↳	-----
DHT sensor library	https://platformio.org/lib/show/19/DHT%20sensor%20library
Adafruit Unified Sensor	https://platformio.org/lib/show/31/Adafruit%20Unified%20Sensor
Blynk	https://platformio.org/lib/show/415/Blynk
ESPAsyncWebServer	https://platformio.org/lib/show/306/ESPAsyncWebServer
Adafruit GFX Library	https://platformio.org/lib/show/13/Adafruit%20GFX%20Library
I2Cdevlib-Core	https://platformio.org/lib/show/11/I2Cdevlib-Core
TimeAlarms	https://platformio.org/lib/show/68/TimeAlarms
PubSubClient	https://platformio.org/lib/show/89/PubSubClient
Timer	https://platformio.org/lib/show/75/Timer
esp8266_mdns	https://platformio.org/lib/show/1091/esp8266_mdns
FEATURED: MONTH	

Name	Url
-----	-----
↳	-----
↳	-----
ArduinoJson	https://platformio.org/lib/show/64/ArduinoJson
DHT sensor library	https://platformio.org/lib/show/19/DHT%20sensor%20library
Adafruit Unified Sensor	https://platformio.org/lib/show/31/Adafruit%20Unified%20Sensor
PubSubClient	https://platformio.org/lib/show/89/PubSubClient
OneWire	https://platformio.org/lib/show/1/OneWire
ESPAsyncTCP	https://platformio.org/lib/show/305/ESPAsyncTCP
Time	https://platformio.org/lib/show/44/Time
DallasTemperature	https://platformio.org/lib/show/54/DallasTemperature
ESPAsyncWebServer	https://platformio.org/lib/show/306/ESPAsyncWebServer
WifiManager	https://platformio.org/lib/show/567/WifiManager

platformio lib uninstall

Contents

- *platformio lib uninstall*
 - *Usage*
 - *Description*
 - *Storage Options*
 - *Examples*

Usage

```
platformio lib [STORAGE_OPTIONS] uninstall [LIBRARY...]
pio lib [STORAGE_OPTIONS] uninstall [LIBRARY...]

# uninstall project dependent library
# (run it from a project root where is located "platformio.ini")
platformio lib uninstall [LIBRARY...]

# uninstall library from global storage
platformio lib --global uninstall [LIBRARY...]
platformio lib -g uninstall [LIBRARY...]

# uninstall library from custom storage
platformio lib --storage-dir /path/to/dir uninstall [LIBRARY...]
platformio lib -d /path/to/dir uninstall [LIBRARY...]

# [LIBRARY...] forms
platformio lib [STORAGE_OPTIONS] uninstall <id>
platformio lib [STORAGE_OPTIONS] uninstall <id>@<version>
platformio lib [STORAGE_OPTIONS] uninstall <id>@<version range>
platformio lib [STORAGE_OPTIONS] uninstall <name>
platformio lib [STORAGE_OPTIONS] uninstall <name>@<version>
platformio lib [STORAGE_OPTIONS] uninstall <name>@<version range>
```

Description

Uninstall specified library

The version supports Semantic Versioning (`<major>.<minor>.<patch>`) and can take any of the following forms:

- `0.1.2` - an exact version number. Use only this exact version
- `^0.1.2` - any compatible version (exact version for `0.x.x` versions)
- `~0.1.2` - any version with the same major and minor versions, and an equal or greater patch version
- `>0.1.2` - any version greater than `0.1.2`. `>=`, `<`, and `<=` are also possible
- `>0.1.0, !=0.2.0, <0.3.0` - any version greater than `0.1.0`, not equal to `0.2.0` and less than `0.3.0`

Storage Options

See base options for [Library Manager](#).

Examples

```
> platformio lib -g uninstall AsyncMqttClient

Library Storage: /storage/dir/...
Uninstalling AsyncMqttClient @ 0.2.0: [OK]
```

platformio lib update

Contents

- [*platformio lib update*](#)
 - [*Usage*](#)
 - [*Description*](#)
 - [*Storage Options*](#)
 - [*Options*](#)
 - [*Examples*](#)

Usage

```
platformio lib [STORAGE_OPTIONS] update [OPTIONS]
pio lib [STORAGE_OPTIONS] update [OPTIONS]

# update all project libraries
# (run it from a project root where is located "platformio.ini")
platformio lib update [OPTIONS]

# update project dependent library
platformio lib [STORAGE_OPTIONS] update [OPTIONS] [LIBRARY...]

# update library in global storage
platformio lib --global update [OPTIONS] [LIBRARY...]
platformio lib -g update [OPTIONS] [LIBRARY...]

# update library in custom storage
platformio lib --storage-dir /path/to/dir update [OPTIONS] [LIBRARY...]
platformio lib -d /path/to/dir update [OPTIONS] [LIBRARY...]

# [LIBRARY...] forms
platformio lib [STORAGE_OPTIONS] update <id>
platformio lib [STORAGE_OPTIONS] update <id>@<version>
platformio lib [STORAGE_OPTIONS] update <id>@<version range>
platformio lib [STORAGE_OPTIONS] update <name>
platformio lib [STORAGE_OPTIONS] update <name>@<version>
platformio lib [STORAGE_OPTIONS] update <name>@<version range>
```

Description

Check or update installed libraries.

The version supports Semantic Versioning (`<major>.<minor>.<patch>`) and can take any of the following forms:

- `0.1.2` - an exact version number. Use only this exact version
- `^0.1.2` - any compatible version (exact version for `0.x.x` versions)

- `~0.1.2` - any version with the same major and minor versions, and an equal or greater patch version
- `>0.1.2` - any version greater than `0.1.2`. `>=`, `<`, and `<=` are also possible
- `>0.1.0, !=0.2.0, <0.3.0` - any version greater than `0.1.0`, not equal to `0.2.0` and less than `0.3.0`

Storage Options

See base options for [Library Manager](#).

Options

-c, --only-check

Do not update, only check for new version

--json-output

Return the output in [JSON](#) format

Examples

1. Update all installed libraries in global storage

```
> platformio lib -g update

Library Storage: /storage/dir/...
Updating ESP8266_SSD1306 @ 3.2.3: [Up-to-date]
Updating EngduinoMagnetometer @ 3.1.0: [Up-to-date]
Updating IRremote @ 2.2.1: [Up-to-date]
Updating Json @ 5.4.0: [Out-of-date]
LibraryManager: Installing id=64 @ 5.6.4
Downloading [########################################] 100%
Unpacking [########################################] 100%
Json @ 5.6.4 has been successfully installed!
Updating PJON @ 1fb26fd: [Checking]
git version 2.7.4 (Apple Git-66)
Already up-to-date.
Updating TextLCD @ 308d188a2d3a: [Checking]
Mercurial Distributed SCM (version 3.8.4)
(see https://mercurial-scm.org for more information)

Copyright (C) 2005-2016 Matt Mackall and others
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
pulling from https://developer.mbed.org/users/simon/code/TextLCD/
searching for changes
no changes found
```

2. Update specified libraries in global storage

```
> platformio lib -g update Json 4

Library Storage: /storage/dir/...
Updating Json @ 5.6.4: [Up-to-date]
Updating IRremote @ 2.2.1: [Up-to-date]
```

Platform Manager

To print all available commands and options use:

```
platformio platform --help
platformio platform COMMAND --help
```

platformio platform frameworks

Contents

- *platformio platform frameworks*
 - *Usage*
 - *Description*
 - * *Options*
 - *Examples*

Usage

```
platformio platform frameworks QUERY [OPTIONS]
pio platform frameworks QUERY [OPTIONS]
```

Description

List supported *Frameworks* (SDKs, etc).

Options

--json-output

Return the output in **JSON** format

Examples

Print all supported frameworks, SDKs, etc.

```
> platformio platform frameworks

arduino ~ Arduino
=====
Arduino Wiring-based Framework allows writing cross-platform software to control_
`-devices attached to a wide range of Arduino boards to create all kinds of creative_
`-coding, interactive objects, spaces or physical experiences.
```

(continues on next page)

(continued from previous page)

Home: <https://platformio.org/frameworks/arduino>

artik-sdk ~ ARTIK SDK

=====

ARTIK SDK **is** a C/C++ SDK targeting Samsung ARTIK platforms. It exposes a **set** of APIs **to ease up development of applications**. These APIs cover hardware buses such **as** **GPIO, SPI, I2C, UART**, connectivity links like Wi-Fi, Bluetooth, Zigbee, **and** network **protocols such as** **HTTP, Websockets, MQTT, and others**.

Home: <https://platformio.org/frameworks/artik-sdk>

cmsis ~ CMSIS

=====

The ARM Cortex Microcontroller Software Interface Standard (CMSIS) **is** a vendor-independent hardware abstraction layer **for** the Cortex-M processor series **and** specifies debugger interfaces. The CMSIS enables consistent **and** simple software interfaces to the processor **for** interface peripherals, real-time operating systems, **and** middleware. It simplifies software re-use, reducing the learning curve **for** new microcontroller developers **and** cutting the time-to-market **for** devices.

Home: <https://platformio.org/frameworks/cmsis>

energia ~ Energia

=====

Energia Wiring-based framework enables pretty much anyone to start easily creating microcontroller-based projects **and** applications. Its easy-to-use libraries **and** functions provide developers of **all** experience levels to start blinking LEDs, buzzing buzzers **and** sensing sensors more quickly than ever before.

Home: <https://platformio.org/frameworks/energia>

espidf ~ ESP-IDF

=====

Espressif IoT Development Framework. Official development framework **for** ESP32.

Home: <https://platformio.org/frameworks/espidf>

libopencm3 ~ libOpenCM3

=====

The libOpenCM3 framework aims to create a free/libre/open-source firmware library **for** various ARM Cortex-M0(+) / M3 / M4 microcontrollers, including ST STM32, TI Tiva **and** Stellaris, NXP LPC **11xx, 13xx, 15xx, 17xx** parts, Atmel SAM3, Energy Micro EFM32 **and** others.

Home: <https://platformio.org/frameworks/libopencm3>

mbed ~ mbed

=====

The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive **and** concise, yet powerful enough to build complex projects. It **is** built on the low-level ARM CMSIS APIs, allowing you to code down to the metal **if** needed. In addition to RTOS, USB **and** Networking libraries, a cookbook of hundreds of reusable peripheral **and** module libraries have been built on top of the SDK by the mbed Developer Community.

Home: <https://platformio.org/frameworks/mbed>

(continues on next page)

(continued from previous page)

```
pumbaa ~ Pumbaa
=====
Pumbaa is Python on top of Simba. The implementation is a port of MicroPython, ↵
→designed for embedded devices with limited amount of RAM and code memory.

Home: https://platformio.org/frameworks/pumbaa

simba ~ Simba
=====
Simba is an RTOS and build framework. It aims to make embedded programming easy and ↵
→portable.

Home: https://platformio.org/frameworks/simba

spl ~ SPL
=====
The ST Standard Peripheral Library provides a set of functions for handling the ↵
→peripherals on the STM32 Cortex-M3 family. The idea is to save the user (the new ↵
→user, in particular) having to deal directly with the registers.

Home: https://platformio.org/frameworks/spl

wiringpi ~ WiringPi
=====
WiringPi is a GPIO access library written in C for the BCM2835 used in the Raspberry ↵
→Pi. ↵
→It's designed to be familiar to people who have used the Arduino "wiring" system.

Home: https://platformio.org/frameworks/wiringpi
```

platformio platform install

Contents

- *platformio platform install*
 - *Usage*
 - *Options*
 - *Description*
 - *Version control*
 - * *Git*
 - * *Mercurial*
 - * *Subversion*
 - *Examples*

Usage

```
platformio platform install [OPTIONS] [PLATFORM...]
pio platform install [OPTIONS] [PLATFORM...]

# [PLATFORM...] forms
platformio platform install <name>
platformio platform install <name>@<version>
platformio platform install <name>@<version range>
platformio platform install <zip or tarball url>
platformio platform install file://<zip or tarball file>
platformio platform install file://<folder>
platformio platform install <repository>
platformio platform install <name=repository> (name it should have locally)
platformio platform install <repository#tag> ("tag" can be commit, branch or tag)
```

Options

--with-package

Install specified package (or alias)

--without-package

Do not install specified package (or alias)

--skip-default

Skip default packages

-f, --force

Reinstall/redownload development platform and its packages if they exist

Description

Install *Development Platforms* and dependent packages.

The version supports Semantic Versioning (<major>.<minor>.<patch>) and can take any of the following forms:

- 0.1.2 - an exact version number. Use only this exact version
- ^0.1.2 - any compatible version (exact version for 0.x.x versions)
- ~0.1.2 - any version with the same major and minor versions, and an equal or greater patch version
- >0.1.2 - any version greater than 0.1.2. >=, <, and <= are also possible
- >0.1.0, !=0.2.0, <0.3.0 - any version greater than 0.1.0, not equal to 0.2.0 and less than 0.3.0

Also, PlatformIO supports installing from local directory or archive. Need to use `file://` prefix before local path. Also, directory or archive should contain `platform.json` manifest.

- `file:///local/path/to/the/platform/dir`
- `file:///local/path/to/the/platform.zip`
- `file:///local/path/to/the/platform.tar.gz`

Version control

PlatformIO supports installing from Git, Mercurial and Subversion, and detects the type of VCS using url prefixes: “git+”, “hg+”, or “svn+”.

Note: PlatformIO requires a working VCS command on your path: git, hg or svn.

Git

The supported schemes are: git, git+https and git+ssh. Here are the supported forms:

- platformio/platform-NAME (short version for GitHub repository)
- <https://github.com/platformio/platform-NAME.git>
- git+git://git.server.org/my-platform
- git+https://git.server.org/my-platform
- git+ssh://git.server.org/my-platform
- git+ssh://user@git.server.org/my-platform
- [user@]host.xz:path/to/repo.git

Passing branch names, a commit hash or a tag name is possible like so:

- <https://github.com/platformio/platform-name.git#master>
- git+git://git.server.org/my-platform#master
- git+https://git.server.org/my-platform#v1.0
- git+ssh://git.server.org/my-platform#7846d8ad52f983f2f2887bdc0f073fe9755a806d

Mercurial

The supported schemes are: hg+http, hg+https and hg+ssh. Here are the supported forms:

- hg+hg://hg.server.org/my-platform
- hg+https://hg.server.org/my-platform
- hg+ssh://hg.server.org/my-platform

Passing branch names, a commit hash or a tag name is possible like so:

- hg+hg://hg.server.org/my-platform#master
- hg+https://hg.server.org/my-platform#v1.0
- hg+ssh://hg.server.org/my-platform#4cfe2fa00668

Subversion

The supported schemes are: svn, svn+svn, svn+http, svn+https and svn+ssh. Here are the supported forms:

- svn+svn://svn.server.org/my-platform

- svn+https://svn.server.org/my-platform
- svn+ssh://svn.server.org/my-platform

You can also give specific revisions to an SVN URL, like so:

- svn+svn://svn.server.org/my-platform#13

Examples

1. Install *Atmel AVR* with default packages

```
> platformio platform install atmelavr

PlatformManager: Installing atmelavr
Downloading...
Unpacking [########################################] 100%
atmelavr @ 0.0.0 has been successfully installed!
PackageManager: Installing tool-scons @ >=2.3.0,<2.6.0
Downloading [########################################] 100%
Unpacking [########################################] 100%
tool-scons @ 2.4.1 has been successfully installed!
PackageManager: Installing toolchain-atmelavr @ ~1.40801.0
Downloading [########################################] 100%
Unpacking [########################################] 100%
toolchain-atmelavr @ 1.40801.0 has been successfully installed!
The platform 'atmelavr' has been successfully installed!
The rest of packages will be installed automatically depending on your build
→environment.
```

2. Install *Atmel AVR* with uploader utility only and skip default packages

```
> platformio platform install atmelavr --skip-default-package --with-package=uploader

PlatformManager: Installing atmelavr
Downloading [########################################] 100%
Unpacking [########################################] 100%
atmelavr @ 0.0.0 has been successfully installed!
PackageManager: Installing tool-micronucleus @ ~1.200.0
Downloading [########################################] 100%
Unpacking [########################################] 100%
tool-micronucleus @ 1.200.0 has been successfully installed!
PackageManager: Installing tool-avrdude @ ~1.60001.0
Downloading [########################################] 100%
Unpacking [########################################] 100%
tool-avrdude @ 1.60001.1 has been successfully installed!
The platform 'atmelavr' has been successfully installed!
The rest of packages will be installed automatically depending on your build
→environment.
```

3. Install the latest development *Atmel AVR* from Git repository

```
> platformio platform install https://github.com/platformio/platform-atmelavr.git

PlatformManager: Installing platform-atmelavr
git version 2.7.4 (Apple Git-66)
Cloning into '/Volumes/MEDIA/tmp/pio3_test_projects/arduino-digihead-master/home_dir/
→platforms/installing-U3ucN0-package'...
```

(continues on next page)

(continued from previous page)

```

remote: Counting objects: 176, done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 176 (delta 114), reused 164 (delta 109), pack-reused 0
Receiving objects: 100% (176/176), 38.86 KiB | 0 bytes/s, done.
Resolving deltas: 100% (114/114), done.
Checking connectivity... done.
Submodule 'examples/arduino-external-libs/lib/OneWire' (https://github.com/
  ↵PaulStoffregen/OneWire.git) registered for path 'examples/arduino-external-libs/lib/
  ↵OneWire'
Cloning into 'examples/arduino-external-libs/lib/OneWire'...
remote: Counting objects: 91, done.
remote: Total 91 (delta 0), reused 0 (delta 0), pack-reused 91
Unpacking objects: 100% (91/91), done.
Checking connectivity... done.
Submodule path 'examples/arduino-external-libs/lib/OneWire': checked out
  ↵'57c18c6de80c13429275f70875c7c341f1719201'
atmelavr @ 0.0.0 has been successfully installed!
PackageManager: Installing tool-scons @ >=2.3.0,<2.6.0
Downloading [#####] 100%
Unpacking [#####] 100%
tool-scons @ 2.4.1 has been successfully installed!
PackageManager: Installing toolchain-atmelavr @ ~1.40801.0
Downloading [#####] 100%
Unpacking [#####] 100%
toolchain-atmelavr @ 1.40801.0 has been successfully installed!
The platform 'https://github.com/platformio/platform-atmelavr.git' has been
  ↵successfully installed!
The rest of packages will be installed automatically depending on your build
  ↵environment.

```

platformio platform list

Contents

- *platformio platform list*
 - *Usage*
 - *Description*
 - * *Options*
 - *Examples*

Usage

```

platformio platform list [OPTIONS]
pio platform list [OPTIONS]

```

Description

List installed *Development Platforms*

Options

--json-output

Return the output in **JSON** format

Examples

```
> platformio platform list

atmelavr ~ Atmel AVR
=====
Atmel AVR 8- and 32-bit MCUs deliver a unique combination of performance, power,
efficiency and design flexibility. Optimized to speed time to market-and easily
adapt to new ones-they are based on the industrys most code-efficient architecture,
for C and assembly programming.

Home: https://platformio.org/platforms/atmelavr
Packages: toolchain-atmelavr, framework-simba
Version: 0.0.0

atmelsam ~ Atmel SAM
=====
Atmel | SMART offers Flash- based ARM products based on the ARM Cortex-M0+, Cortex-M3,
and Cortex-M4 architectures, ranging from 8KB to 2MB of Flash including a rich
peripheral and feature mix.

Home: https://platformio.org/platforms/atmelsam
Packages: framework-arduinomas, framework-mbed, framework-simba, toolchain-
gccarmnoneabi, tool-bossac
Version: 0.0.0

espressif8266 ~ Espressif 8266
=====
Espressif Systems is a privately held fabless semiconductor company. They provide,
wireless communications and Wi-Fi chips which are widely used in mobile devices and
the Internet of Things applications.

Home: https://platformio.org/platforms/espressif8266
Packages: framework-simba, tool-esptool, framework-arduinoespressif8266, sdk-esp8266,
toolchain-xtensa
Version: 0.0.0
...
```

platformio platform search

Contents

- *platformio platform search*
 - *Usage*
 - *Description*
 - * *Options*
 - *Examples*

Usage

```
platformio platform search QUERY [OPTIONS]
pio platform search QUERY [OPTIONS]
```

Description

Search for development *Development Platforms*

Options

--json-output

Return the output in **JSON** format

Examples

1. Print all available development platforms

```
> platformio platform search

atmelavr ~ Atmel AVR
=====
Atmel AVR 8- and 32-bit MCUs deliver a unique combination of performance, power,
efficiency and design flexibility. Optimized to speed time to market-and easily,
adapt to new ones-they are based on the industrys most code-efficient architecture,
for C and assembly programming.

Home: https://platformio.org/platforms/atmelavr
Packages: toolchain-atmelavr, framework-arduinoavr, framework-simba, tool-avrdude,
tool-micronucleus

atmelsam ~ Atmel SAM
=====
Atmel | SMART offers Flash- based ARM products based on the ARM Cortex-M0+, Cortex-M3,
and Cortex-M4 architectures, ranging from 8KB to 2MB of Flash including a rich,
peripheral and feature mix.

Home: https://platformio.org/platforms/atmelsam
Packages: toolchain-gccarmnoneabi, framework-arduinomas, framework-simba, tool-
openocd, framework-mbed, tool-avrdude, tool-bossac
```

(continues on next page)

(continued from previous page)

```
espressif32 ~ Espressif 32
=====
Espressif Systems is a privately held fabless semiconductor company. They provide
↳ wireless communications and Wi-Fi chips which are widely used in mobile devices and
↳ the Internet of Things applications.

Home: https://platformio.org/platforms/espressif32
Packages: toolchain-xtensa32, framework-simba, framework-arduinoespressif32, ↳
↳ framework-pumbaa, framework-espifdf, tool-esptoolpy

espressif8266 ~ Espressif 8266
=====
Espressif Systems is a privately held fabless semiconductor company. They provide
↳ wireless communications and Wi-Fi chips which are widely used in mobile devices and
↳ the Internet of Things applications.

Home: https://platformio.org/platforms/espressif8266
Packages: toolchain-xtensa, framework-simba, tool-esptool, tool-mkspiffs, tool-
↳ espotapy, framework-arduinoespressif8266, sdk-esp8266

freescalekinetis ~ Freescale Kinetis
=====
Freescale Kinetis Microcontrollers is family of multiple hardware- and software-
↳ compatible ARM Cortex-M0+, Cortex-M4 and Cortex-M7-based MCU series. Kinetis MCUs
↳ offer exceptional low-power performance, scalability and feature integration.
...
```

2. Search for TI development platforms

```
> platformio platform search texas

timsp430 ~ TI MSP430
=====
MSP430 microcontrollers (MCUs) from Texas Instruments (TI) are 16-bit, RISC-based,
↳ mixed-signal processors designed for ultra-low power. These MCUs offer the lowest
↳ power consumption and the perfect mix of integrated peripherals for thousands of
↳ applications.

Home: https://platformio.org/platforms/timsp430
Packages: toolchain-timsp430, tool-mspdebug, framework-energiamsp430, framework-
↳ arduinomsp430

titiva ~ TI TIVA
=====
Texas Instruments TM4C12x MCUs offer the industry's most popular ARM Cortex-M4 core
↳ with scalable memory and package options, unparalleled connectivity peripherals,
↳ advanced application functions, industry-leading analog integration, and extensive
↳ software solutions.

Home: https://platformio.org/platforms/titiva
Packages: ldscripts, framework-libopencm3, toolchain-gccarmnoneabi, tool-lm4flash, ↳
↳ framework-energiativa
```

```
> platformio platform search framework-mbed
```

(continues on next page)

(continued from previous page)

```

atmelsam ~ Atmel SAM
=====
Atmel | SMART offers Flash- based ARM products based on the ARM Cortex-M0+, Cortex-M3
→and Cortex-M4 architectures, ranging from 8KB to 2MB of Flash including a rich
→peripheral and feature mix.

Home: https://platformio.org/platforms/atmelsam
Packages: toolchain-gccarmnoneabi, framework-arduinomas, framework-simba, tool-
→openocd, framework-mbed, ldscripts, tool-bossac

freescalekinetis ~ Freescale Kinetis
=====
Freescale Kinetis Microcontrollers is family of multiple hardware- and software-
→compatible ARM Cortex-M0+, Cortex-M4 and Cortex-M7-based MCU series. Kinetis MCUs
→offer exceptional low-power performance, scalability and feature integration.

Home: https://platformio.org/platforms/freescalekinetis
Packages: framework-mbed, toolchain-gccarmnoneabi

nordicnrf51 ~ Nordic nRF51
=====
The Nordic nRF51 Series is a family of highly flexible, multi-protocol, system-on-
→chip (SoC) devices for ultra-low power wireless applications. nRF51 Series devices
→support a range of protocol stacks including Bluetooth Smart (previously called
→Bluetooth low energy), ANT and proprietary 2.4GHz protocols such as Gazell.

Home: https://platformio.org/platforms/nordicnrf51
Packages: framework-mbed, tool-rfdloader, toolchain-gccarmnoneabi, framework-
→arduinonordicnrf51

nxplpc ~ NXP LPC
=====
The NXP LPC is a family of 32-bit microcontroller integrated circuits by NXP
→Semiconductors. The LPC chips are grouped into related series that are based around
→the same 32-bit ARM processor core, such as the Cortex-M4F, Cortex-M3, Cortex-M0+,
→or Cortex-M0. Internally, each microcontroller consists of the processor core,
→static RAM memory, flash memory, debugging interface, and various peripherals.

Home: https://platformio.org/platforms/nxplpc
Packages: framework-mbed, toolchain-gccarmnoneabi

siliconlabsefm32 ~ Silicon Labs EFM32
=====
Silicon Labs EFM32 Gecko 32-bit microcontroller (MCU) family includes devices that
→offer flash memory configurations up to 256 kB, 32 kB of RAM and CPU speeds up to
→48 MHz. Based on the powerful ARM Cortex-M core, the Gecko family features
→innovative low energy techniques, short wake-up time from energy saving modes and a
→wide selection of peripherals, making it ideal for battery operated applications
→and other systems requiring high performance and low-energy consumption.

Home: https://platformio.org/platforms/siliconlabsefm32
Packages: framework-mbed, toolchain-gccarmnoneabi

ststm32 ~ ST STM32
=====
The STM32 family of 32-bit Flash MCUs based on the ARM Cortex-M processor is designed
→to offer new degrees of freedom to MCU users. It offers a 32-bit product range that
→combines very high performance, real-time capabilities, digital signal processing,(continues on next page)
→and low-power, low-voltage operation, while maintaining full integration and ease
→of development.
```

(continued from previous page)

```
Home: https://platformio.org/platforms/ststm32
Packages: framework-libopencm3, toolchain-gccarmnoneabi, tool-stlink, framework-spl,
→framework-cmsis, framework-mbed, ldscripts

teensy ~ Teensy
=====
Teensy is a complete USB-based microcontroller development system, in a very small
→footprint, capable of implementing many types of projects. All programming is done
→via the USB port. No special programmer is needed, only a standard USB cable and a
→PC or Macintosh with a USB port.

Home: https://platformio.org/platforms/teensy
Packages: framework-arduinoteensy, tool-teensy, toolchain-gccarmnoneabi, framework-
→mbed, toolchain-atmelavr, ldscripts
...
```

platformio platform show

Contents

- *platformio platform show*
 - *Usage*
 - *Description*
 - *Examples*

Usage

```
platformio platform show PLATFORM
pio platform show PLATFORM
```

Description

Show details about *Development Platforms*

Examples

```
> platformio platform show atmelavr

atmelavr ~ Atmel AVR
=====
Atmel AVR 8- and 32-bit MCUs deliver a unique combination of performance, power
→efficiency and design flexibility. Optimized to speed time to market-and easily
→adapt to new ones-they are based on the industrys most code-efficient architecture
→for C and assembly programming.
```

(continues on next page)

(continued from previous page)

```
Version: 1.2.1
Home: https://platformio.org/platforms/atmelavr
License: Apache-2.0
Frameworks: simba, arduino

Package toolchain-atmelavr
-----
Type: toolchain
Requirements: ~1.40902.0
Installed: Yes
Description: avr-gcc
Url: http://www.atmel.com/products/microcontrollers/avr/32-bitavruc3.aspx?tab=tools
Version: 1.40902.0 (4.9.2)

Package framework-arduinoavr
-----
Type: framework
Requirements: ~1.10612.1
Installed: Yes
Url: https://www.arduino.cc/en/Main/Software
Version: 1.10612.1 (1.6.12)
Description: Arduino Wiring-based Framework (AVR Core, 1.6)

Package framework-simba
-----
Type: framework
Requirements: >=7.0.0
Installed: Yes
Url: https://github.com/eerimoq/simba
Version: 11.0.0
Description: Simba Embedded Programming Platform

Package tool-avrdude
-----
Type: uploader
Requirements: ~1.60300.0
Installed: Yes
Description: AVRDUDE
Url: http://www.nongnu.org/avrdude/
Version: 1.60300.0 (6.3.0)

Package tool-micronucleus
-----
Type: uploader
Requirements: ~1.200.0
Installed: No (optional)
```

platformio platform uninstall

Contents

- [platformio platform uninstall](#)

- *Usage*
- *Description*
- *Examples*

Usage

```
platformio platform uninstall [PLATFORM...]
pio platform uninstall [PLATFORM...]

# uninstall specific platform version using Semantic Versioning
platformio platform uninstall PLATFORM@X.Y.Z
```

Description

Uninstall specified *Development Platforms*

Examples

```
> platformio platform uninstall atmelavr
Uninstalling platform atmelavr @ 0.0.0:      [OK]
Uninstalling package tool-scons @ 2.4.1:      [OK]
Uninstalling package toolchain-atmelavr @ 1.40801.0:      [OK]
The platform 'atmelavr' has been successfully uninstalled!
```

platformio platform update

Contents

- *platformio platform update*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio platform update [OPTIONS] [PLATFORM...]
pio platform update [OPTIONS] [PLATFORM...]

# update specific platform version using Semantic Versioning
platformio platform update PLATFORM@X.Y.Z
```

Description

Check or update installed *Development Platforms*

Options

-p, --only-packages

Update only the platform related packages. Do not update development platform build scripts, board configs and etc.

-c, --only-check

Do not update, only check for a new version

--json-output

Return the output in **JSON** format

Examples

```
> platformio platform update

Platform atmelavr
-----
Updating atmelavr @ 0.0.0: [Up-to-date]
Updating framework-arduinoavr @ 1.10608.1: [Up-to-date]
Updating tool-avrdude @ 1.60001.1: [Up-to-date]
Updating toolchain-atmelavr @ 1.40801.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform espressif8266
-----
Updating espressif @ 0.0.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]
Updating toolchain-xtensa @ 1.40802.0: [Up-to-date]
Updating tool-esptool @ 1.409.0: [Up-to-date]
Updating tool-mkspiffs @ 1.102.0: [Up-to-date]
Updating framework-arduinoespressif8266 @ 1.20300.0: [Up-to-date]
Updating sdk-esp8266 @ 1.10502.0: [Up-to-date]

Platform teensy
-----
Updating teensy @ 0.0.0: [Up-to-date]
Updating framework-arduinoteensy @ 1.128.0: [Up-to-date]
Updating tool-teensy @ 1.1.0: [Up-to-date]
Updating framework-mbed @ 1.121.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]
Updating toolchain-atmelavr @ 1.40801.0: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]

...
```

platformio remote

Helper command for *PIO Remote*.

To print all available commands and options use:

```
pio remote --help
platformio remote --help
platformio remote COMMAND --help

# run command on the specified PIO Remote Agents
platformio remote --agent NAME_1 --agent NAME_N COMMAND
```

PIO Remote Agent

Start *PIO Remote Agent* on a host machine and work remotely with your devices **WITHOUT** extra software, services, SSH, VPN, tunneling or opening incoming network ports.

PIO Remote supports wired and wireless devices. Wired devices should be connected physically to host machine where *PIO Remote Agent* is started, where wireless devices should be visible for *PIO Remote Agent* to provide network operations Over-The-Air (OTA).

Contents

- *PIO Remote Agent*
 - *platformio remote agent list*
 - * *Usage*
 - * *Description*
 - * *Example*
 - *platformio remote agent start*
 - * *Usage*
 - * *Description*
 - * *Options*
 - *platformio remote agent reload*
 - * *Usage*
 - * *Description*
 - * *Example*

platformio remote agent list

Usage

```
platformio remote agent list
pio remote agent list
```

Description

List active *PIO Remote Agent* s started using own *PIO Account* or shared with you by other PlatformIO developers.

Example

```
> platformio remote agent list

PIO Plus (https://pioplus.com)

innomac.local
-----
ID: 98853d930.....788d77375e7
Started: 2016-10-26 16:32:56
```

platformio remote agent start

Usage

```
platformio remote agent start [OPTIONS]
pio remote agent start [OPTIONS]
```

Description

Start *PIO Remote Agent* and work remotely with your devices from anywhere in the world. This command can be run as daemon or added to autostart list of your OS.

Options

-n, --name

Agent name/alias. By default, machine's hostname will be used. You can use this name later for *platformio remote device* and *platformio remote run* commands. Good names are home, office, lab or etc.

-s, --share

Share your agent/devices with other PlatformIO developers who have *PIO Account*: friends, co-workers, team, etc.

The valid value for *--share* option is E-Mail address that was used for *platformio account register* command.

-d, --working-dir

A working directory where *PIO Remote Agent* stores projects data for incremental synchronization and embedded programs for PIO Process Supervisor.

platformio remote agent reload

Usage

```
platformio remote agent reload
pio remote agent reload

# reload specified PIO Remote Agents
platformio remote --agent NAME reload
```

Description

Allows gracefully reload one or more *PIO Remote Agent* ‘s.

Example

```
> platformio remote agent list

PIO Plus (https://pioplus.com)

innomac.local
-----
ID: 98853d93.....77375e7
Reloaded: 2016-11-11 23:33:32
```

platformio remote device

Remote Device: monitor remote device or list existing.

Contents

- *platformio remote device*
 - *platformio remote device list*
 - * *Usage*
 - * *Description*
 - * *Options*
 - * *Example*
 - *platformio remote device monitor*
 - * *Usage*
 - * *Description*
 - * *Options*
 - * *Examples*

platformio remote device list

Usage

```
platformio remote device list [OPTIONS]
pio remote device list [OPTIONS]

# List devices from the specified agents. Multiple agents are allowed.
platformio remote --agent NAME device list [OPTIONS]
```

Description

List **Serial Ports** on remote machines where *PIO Remote Agent* is started.

You can list devices from the specified remote machines using `--agent NAME` option between “remote” & “device” sub-commands. For example, you have run `platformio remote agent start --name` command with “home” and “office” options:

- `platformio remote agent start --name home`
- `platformio remote agent start --name office`

Now, to list devices from office machine please use `platformio remote --agent office device list`.

Multiple agents are allowed (`platformio remote --agent lab1 --agent lab3 device ...`).

Options

--json-output

Return the output in **JSON** format

Example

```
> platformio remote device list

PIO Plus (https://pioplus.com)

Agent innomac.local
=====
/dev/cu.Bluetooth-Incoming-Port
-----
Hardware ID: n/a
Description: n/a
/dev/cu.obd2ecu-SPPDev
-----
Hardware ID: n/a
Description: n/a
/dev/cu.usbmodemFA1431
-----
Hardware ID: USB VID:PID=2A03:0043 SER=75435353038351015271 LOCATION=250-1.4.3
Description: Arduino Uno
/dev/cu.usbserial-A6004003
```

(continues on next page)

(continued from previous page)

```
-----
Hardware ID: USB VID:PID=0403:6001 SER=A6004003 LOCATION=253-1.3.1
Description: FT232R USB UART - FT232R USB UART
/dev/cu.SLAB_USBtoUART
-----
Hardware ID: USB VID:PID=10C4:EA60 SER=0001 LOCATION=253-1.3.2
Description: CP2102 USB to UART Bridge Controller - CP2102 USB to UART Bridge Controller
/dev/cu.usbmodem589561
-----
Hardware ID: USB VID:PID=16C0:0483 SER=589560 LOCATION=250-1.4.1
Description: USB Serial
```

platformio remote device monitor

Remote Serial Port Monitor

Usage

```
platformio remote device monitor [OPTIONS]
pio remote device monitor [OPTIONS]

# Connect to a specified agent
platformio remote --agent NAME device monitor [OPTIONS]
platformio remote -a NAME device monitor [OPTIONS]
```

Description

Connect to Serial Port of remote device and receive or send data in real time. [PIO Remote Agent](#) should be started before on a remote machine.

To control *monitor* please use these “hot keys”:

- Ctrl+C Quit
- Ctrl+T Menu
- Ctrl+T followed by Ctrl+H Help

Options

-p, --port

Port, a number or a device name

-b, --baud

Set baud rate, default 9600

--parity

Set parity (*None, Even, Odd, Space, Mark*), one of [N, E, O, S, M], default N

--rtscts

Enable RTS/CTS flow control, default Off

--xonxoff

Enable software flow control, default Off

--rts

Set initial RTS line state, default 0

--dtr

Set initial DTR line state, default 0

--echo

Enable local echo, default Off

--encoding

Set the encoding for the serial port (e.g. hexlify, Latin1, UTF-8), default UTF-8.

-f, --filter

Add text transformation. Available filters:

- `colorize` Apply different colors for received and echo
- `debug` Print what is sent and received
- `default` Remove typical terminal control codes from input
- `direct` Do-nothing: forward all data unchanged
- `nocontrol` Remove all control codes, incl. CR+LF
- `printable` Show decimal code for all non-ASCII characters and replace most control codes

--eol

End of line mode (CR, LF or CRLF), default CRLF

--raw

Do not apply any encodings/transformations

--exit-char

ASCII code of special character that is used to exit the application, default 3 (DEC, Ctrl+C).

For example, to use Ctrl+] run `platformio remote device monitor --exit-char 29.`

--menu-char

ASCII code of special character that is used to control miniterm (menu), default 20 (DEC)

--quiet

Diagnostics: suppress non-error messages, default Off

Examples

1. Show available options for *monitor*

```
> platformio remote device monitor --help

Usage: platformio remote device monitor [OPTIONS]

Options:
  -p, --port TEXT          Port, a number or a device name
  -b, --baud INTEGER       Set baud rate, default=9600
  --parity [N|E|O|S|M]     Set parity, default=N
  --rtscts                 Enable RTS/CTS flow control, default=Off
  --xonxoff                Enable software flow control, default=Off
  --rts [0|1]                Set initial RTS line state, default=0
  --dtr [0|1]                Set initial DTR line state, default=0
  --echo                   Enable local echo, default=Off
  --encoding TEXT           Set the encoding for the serial port (e.g. hexlify,
                           Latin1, UTF-8), default: UTF-8
  -f, --filter TEXT         Add text transformation
  --eol [CR|LF|CRLF]        End of line mode, default=CRLF
  --raw                     Do not apply any encodings/transformations
  --exit-char INTEGER        ASCII code of special character that is used to exit
                           the application, default=29 (DEC)
  --menu-char INTEGER        ASCII code of special character that is used to
                           control miniterm (menu), default=20 (DEC)
  --quiet                  Diagnostics: suppress non-error messages, default=Off
  -h, --help                 Show this message and exit.
```

2. Communicate with serial device and print help inside terminal

```
> platformio remote device monitor

--- Available ports:
--- /dev/cu.Bluetooth-Incoming-Port n/a
--- /dev/cu.Bluetooth-Modem n/a
--- /dev/cu.SLAB_USBtoUART CP2102 USB to UART Bridge Controller
--- /dev/cu.obd2ecu-SPPDev n/a
Enter port name:/dev/cu.SLAB_USBtoUART
--- Miniterm on /dev/cu.SLAB_USBtoUART: 9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Hello PlatformIO!
---
--- Ctrl+] Exit program
--- Ctrl+T Menu escape key, followed by:
--- Menu keys:
---   Ctrl+T Send the menu character itself to remote
---   Ctrl+] Send the exit character itself to remote
---   Ctrl+I Show info
---   Ctrl+U Upload file (prompt will be shown)
--- Toggles:
---   Ctrl+R RTS          Ctrl+E local echo
---   Ctrl+D DTR          Ctrl+B BREAK
---   Ctrl+L line feed    Ctrl+A Cycle repr mode
---
--- Port settings (Ctrl+T followed by the following):
---   p      change port
---   7 8    set data bits
---   n e o s m  change parity (None, Even, Odd, Space, Mark)
---   1 2 3    set stop bits (1, 2, 1.5)
---   b      change baud rate
```

(continues on next page)

(continued from previous page)

```
--- x X      disable/enable software flow control
--- r R      disable/enable hardware flow control
--- exit ---
```

platformio remote run

Remote Firmware Updates

Contents

- *platformio remote run*
 - *Usage*
 - *Description*
 - *Options*
 - *Example*

Usage

```
platformio remote run [OPTIONS]
pio remote run [OPTIONS]

# process environments using specified PIO Remote Agent
platformio remote --agent NAME run [OPTIONS]
```

Description

Process remotely environments which are defined in *Project Configuration File platformio.ini* file. By default, *PIO Remote* builds project on a host machine and deploy final firmware (program) to a remote device (embedded board).

If you need to process project on a remote machine, please use *platformio remote run --force-remote* option. In this case, *PIO Remote* will automatically synchronize your project with remote machine, install required toolchains, frameworks, SDKs, etc., and process project.

Options

-e, --environment

Process specified environments.

You can also specify which environments should be processed by default using *env_default* option from *Project Configuration File platformio.ini*.

-t, --target

Process specified targets.

Built-in targets:

- `clean` delete compiled object files, libraries and firmware/program binaries
- `upload` firmware “auto-uploading” for embedded platforms
- `program` firmware “auto-uploading” for embedded platforms using external programmer (available only for *Atmel AVR*)
- `buildfs` *Uploading files to file system SPIFFS*
- `uploadfs` *Uploading files to file system SPIFFS*
- `envdump` dump current build environment
- `size` print the size of the sections in a firmware/program

--upload-port

Custom upload port of embedded board. To print all available ports use `platformio remote device` command.

If upload port is not specified, PlatformIO will try to detect it automatically.

-d, --project-dir

Specify the path to project directory. By default, `--project-dir` is equal to current working directory (CWD).

-v, --verbose

Shows detailed information when processing environments.

This option can be set globally using `force_verbose` setting or by environment variable `PLATFORMIO_SETTING_FORCE_VERBOSE`.

--disable-auto-clean

Disable auto-clean of `build_dir` when `Project Configuration File platformio.ini` or `src_dir` (project structure) have been modified.

-r, --force-remote

By default, `PIO Remote` builds project on a host machine and deploy final firmware (program) to remote device (embedded board).

If you need to process project on remote machine, please use `platformio remote run --force-remote` option. In this case, `PIO Remote` will automatically synchronize your project with remote machine, install required toolchains, frameworks, SDKs, etc., and process project.

Example

```
> platformio remote run --environment uno --target upload

PIO Plus (https://pioplus.com)
Building project locally
[Wed Oct 26 16:35:09 2016] Processing uno (platform: atmelavr, board: uno, framework:arduino)
-----
Verbose mode can be enabled via `'-v, --verbose` option
Collected 25 compatible libraries
Looking for dependencies...
Project does not have dependencies
Compiling .pioenvs/uno/src/main.o
Archiving .pioenvs/uno/libFrameworkArduinoVariant.a
Indexing .pioenvs/uno/libFrameworkArduinoVariant.a
Compiling .pioenvs/uno/FrameworkArduino/CDC.o
```

(continues on next page)

(continued from previous page)

```

Compiling .pioenvs/uno/FrameworkArduino/HardwareSerial.o
Compiling .pioenvs/uno/FrameworkArduino/HardwareSerial0.o
Compiling .pioenvs/uno/FrameworkArduino/HardwareSerial1.o
Compiling .pioenvs/uno/FrameworkArduino/HardwareSerial2.o
Compiling .pioenvs/uno/FrameworkArduino/HardwareSerial3.o
Compiling .pioenvs/uno/FrameworkArduino/IPAddress.o
Compiling .pioenvs/uno/FrameworkArduino/PluggableUSB.o
Compiling .pioenvs/uno/FrameworkArduino/Print.o
Compiling .pioenvs/uno/FrameworkArduino/Stream.o
Compiling .pioenvs/uno/FrameworkArduino/Tone.o
Compiling .pioenvs/uno/FrameworkArduino/USBCore.o
Compiling .pioenvs/uno/FrameworkArduino/WIInterrupts.o
Compiling .pioenvs/uno/FrameworkArduino/WMath.o
Compiling .pioenvs/uno/FrameworkArduino/WString.o
Compiling .pioenvs/uno/FrameworkArduino/_wiring_pulse.o
Compiling .pioenvs/uno/FrameworkArduino/abi.o
Compiling .pioenvs/uno/FrameworkArduino/hooks.o
Compiling .pioenvs/uno/FrameworkArduino/main.o
Compiling .pioenvs/uno/FrameworkArduino/new.o
Compiling .pioenvs/uno/FrameworkArduino/wiring.o
Compiling .pioenvs/uno/FrameworkArduino/wiring_analog.o
Compiling .pioenvs/uno/FrameworkArduino/wiring_digital.o
Compiling .pioenvs/uno/FrameworkArduino/wiring_pulse.o
Compiling .pioenvs/uno/FrameworkArduino/wiring_shift.o
Archiving .pioenvs/uno/libFrameworkArduino.a
Indexing .pioenvs/uno/libFrameworkArduino.a
Linking .pioenvs/uno/firmware.elf
Checking program size
Building .pioenvs/uno/firmware.hex
text      data      bss      dec      hex filename
2574        48      168     2790     ae6 .pioenvs/uno/firmware.elf
=====
===== [SUCCESS] Took 10.01 seconds =====
===== [SUMMARY] =====
Environment nodemcuv2      [SKIP]
Environment uno_pic32      [SKIP]
Environment teensy31       [SKIP]
Environment uno            [SUCCESS]
=====
===== [SUCCESS] Took 10.01 seconds =====
Uploading firmware remotely
[Wed Oct 26 19:35:20 2016] Processing uno (platform: atmelavr, board: uno, framework: arduino)
-----
-----
Verbose mode can be enabled via `--v, --verbose` option
Looking for upload port...
Auto-detected: /dev/cu.usbmodemFA1431
Uploading .pioenvs/uno/firmware.hex
avrdude: AVR device initialized and ready to accept instructions
Reading | ##### | 100% 0.00s
avrdude: Device signature = 0x1e950f
avrdude: reading input file ".pioenvs/uno/firmware.hex"
avrdude: writing flash (2622 bytes):
Writing | ##### | 100% 0.43s
avrdude: 2622 bytes of flash written
avrdude: verifying flash memory against .pioenvs/uno/firmware.hex:
avrdude: load data flash data from input file .pioenvs/uno/firmware.hex:
avrdude: input file .pioenvs/uno/firmware.hex contains 2622 bytes

```

(continues on next page)

(continued from previous page)

```
avrdude: reading on-chip flash data:  
Reading | #####|#####|#####|#####|#####|#####|#####|#####|#####|##### | 100% 0.34s  
avrdude: verifying ...  
avrdude: 2622 bytes of flash verified  
avrdude done. Thank you.  
===== [SUCCESS] Took 3.04 seconds =====  
===== [SUMMARY] =====  
Environment nodemcuv2 [SKIP]  
Environment uno_pic32 [SKIP]  
Environment teensy31 [SKIP]  
Environment uno [SUCCESS]  
===== [SUCCESS] Took 3.04 seconds =====
```

platformio remote test

Helper command for remote [PIO Unit Testing](#).

Contents

- *platformio remote test*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio remote test [OPTIONS]  
pio remote test [OPTIONS]  
  
# run tests on specified PIO Remote Agent  
platformio remote --agent NAME test [OPTIONS]
```

Description

Run remotely tests from PlatformIO based project. More details about PlatformIO [PIO Unit Testing](#).

This command allows you to apply the tests for the environments specified in [Project Configuration File platformio.ini](#).

Options

-e, --environment

Process specified environments. More details [platformio run --environment](#)

-i, --ignore

Ignore tests where the name matches specified patterns. More than one pattern is allowed. If you need to ignore some tests for the specific environment, please take a look at `test_ignore` option from *Project Configuration File platformio.ini*.

Pattern	Meaning
*	matches everything
?	matches any single character
[seq]	matches any character in seq
[!seq]	matches any character not in seq

For example, `platformio remote test --ignore "mytest*" -i "test[13]"`

--upload-port

A port that is intended for firmware uploading. To list available ports please use `platformio device list` command.

If upload port is not specified, PlatformIO will try to detect it automatically.

--test-port

A Serial/UART port that PlatformIO uses as communication interface between PlatformIO Unit Test Engine and target device. To list available ports please use `platformio device list` command.

If test port is not specified, PlatformIO will try to detect it automatically.

-d, --project-dir

Specify the path to project directory. By default, `--project-dir` is equal to current working directory (CWD).

-r, --force-remote

By default, `PIO Remote` processes project on a host machine and deploy final testing firmware (program) to remote device (embedded board).

If you need to process project on remote machine, please use `platformio remote test --force-remote` option. In this case, `PIO Remote` will automatically synchronize your project with remote machine, install required toolchains, frameworks, SDKs, etc., and process project.

--without-building

Skip building stage.

--without-uploading

Skip uploading stage

-v, --verbose

Shows detailed information when processing environments.

This option can be set globally using `force_verbose` setting or by environment variable `PLATFORMIO_SETTING_FORCE_VERBOSE`.

Examples

For the examples please follow to [PIO Unit Testing](#) page.

platformio remote update

Contents

- *platformio remote update*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio remote update [OPTIONS]
pio remote update [OPTIONS]

# start update process on the specified agents/machines
platformio remote --agent NAME update [OPTIONS]
```

Description

Check or update installed *Development Platforms* and global *Libraries* on the remote machine.

Options

-c, --only-check

Do not update, only check for new version

Examples

```
> platformio remote update

PIO Plus (https://pioplus.com)

Platform Manager
=====
Platform timsp430
-----
Updating timsp430 @ 0.0.0: [Up-to-date]
Updating toolchain-timsp430 @ 1.40603.0: [Up-to-date]
Updating framework-energiamsp430 @ 1.17.0: [Up-to-date]
Updating framework-arduinomsp430 @ 1.10601.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform freescalekinetis
-----
```

(continues on next page)

(continued from previous page)

```

Updating freescalekinetis @ 0.0.0: [Up-to-date]
Updating framework-mbed @ 1.121.1: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform ststm32
-----
Updating ststm32 @ 0.0.0: [Up-to-date]
Updating framework-libopencm3 @ 1.1.0: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]
Updating tool-stlink @ 1.10200.0: [Up-to-date]
Updating framework-spl @ 1.10201.0: [Up-to-date]
Updating framework-cmsis @ 1.40300.0: [Up-to-date]
Updating framework-mbed @ 1.121.1: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform lattice_ice40
-----
Updating lattice_ice40 @ 0.0.0: [Up-to-date]
Updating toolchain-icestorm @ 1.7.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform atmelavr
-----
Updating atmelavr @ 0.0.0: [Up-to-date]
Updating framework-arduinoavr @ 1.10608.1: [Up-to-date]
Updating tool-avrdude @ 1.60001.1: [Up-to-date]
Updating toolchain-atmelavr @ 1.40801.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform espressif8266
-----
Updating espressif8266 @ 0.0.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]
Updating toolchain-xtensa @ 1.40802.0: [Up-to-date]
Updating tool-esptool @ 1.409.0: [Up-to-date]
Updating tool-mkspiffs @ 1.102.0: [Up-to-date]
Updating framework-arduinoespressif8266 @ 1.20300.0: [Up-to-date]
Updating sdk-esp8266 @ 1.10502.0: [Up-to-date]

Platform linux_x86_64
-----
Updating linux_x86_64 @ 0.0.0: [Up-to-date]
Updating toolchain-gcclinux64 @ 1.40801.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform windows_x86
-----
Updating windows_x86 @ 0.0.0: [Up-to-date]
Updating toolchain-gccmingw32 @ 1.40800.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform teensy
-----
Updating teensy @ 0.0.0: [Up-to-date]
Updating framework-arduinoteensy @ 1.128.0: [Up-to-date]
Updating tool-teensy @ 1.1.0: [Up-to-date]

```

(continues on next page)

(continued from previous page)

```
Updating framework-mbed @ 1.121.1: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]
Updating toolchain-atmelavr @ 1.40801.0: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]

Platform nordicnrf51
-----
Updating nordicnrf51 @ 0.0.0: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]
Updating framework-arduinoonordicnrf51 @ 1.20302.0: [Up-to-date]
Updating framework-mbed @ 1.121.1: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform titiva
-----
Updating titiva @ 0.0.0: [Up-to-date]
Updating framework-libopencm3 @ 1.1.0: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]
Updating framework-energiativa @ 1.17.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform atmelsam
-----
Updating atmelsam @ 0.0.0: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]
Updating tool-openocd @ 1.900.0: [Up-to-date]
Updating framework-mbed @ 1.121.1: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]
Updating tool-avrdude @ 1.60001.1: [Up-to-date]
Updating tool-bossac @ 1.10601.0: [Up-to-date]

Platform siliconlabsefm32
-----
Updating siliconlabsefm32 @ 0.0.0: [Up-to-date]
Updating framework-mbed @ 1.121.1: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform microchippic32
-----
Updating microchippic32 @ 0.0.0: [Up-to-date]
Updating framework-arduinomicrochippic32 @ 1.10201.0: [Up-to-date]
Updating toolchain-microchippic32 @ 1.40803.0: [Up-to-date]
Updating tool-pic32prog @ 1.200200.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform linux_i686
-----
Updating linux_i686 @ 0.0.0: [Up-to-date]
Updating toolchain-gcclinux32 @ 1.40801.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform intel_arc32
-----
Updating intel_arc32 @ 0.0.0: [Up-to-date]
Updating framework-arduinointel @ 1.10006.0: [Up-to-date]
Updating tool-arduino101load @ 1.124.0: [Up-to-date]
```

(continues on next page)

(continued from previous page)

```

Updating toolchain-intelarc32 @ 1.40805.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform nxplpc
-----
Updating nxplpc @ 0.0.0: [Up-to-date]
Updating framework-mbed @ 1.121.1: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform linux_arm
-----
Updating linux_arm @ 0.0.0: [Up-to-date]
Updating toolchain-gccarmlinuxgnueabi @ 1.40802.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform native
-----
Updating native @ 0.0.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Library Manager
=====
Updating Adafruit-GFX @ 334e815bc1: [Up-to-date]
Updating Adafruit-ST7735 @ d53d4bf03a: [Up-to-date]
Updating Adafruit-DHT @ 09344416d2: [Up-to-date]
Updating Adafruit-Unified-Sensor @ f2af6f4efc: [Up-to-date]
Updating ESP8266_SSD1306 @ 3.2.3: [Up-to-date]
Updating EngduinoMagnetometer @ 3.1.0: [Up-to-date]
Updating IRremote @ 2.2.1: [Up-to-date]
Updating Json @ 5.6.4: [Up-to-date]
Updating MODSERIAL @ d8422efe47: [Up-to-date]
Updating PJSON @ 1fb26fd: [Checking]
git version 2.7.4 (Apple Git-66)
Already up-to-date.
Updating Servo @ 36b69a7ced07: [Checking]
Mercurial Distributed SCM (version 3.8.4)
(see https://mercurial-scm.org for more information)

Copyright (C) 2005-2016 Matt Mackall and others
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
pulling from https://developer.mbed.org/users/simon/code/Servo/
searching for changes
no changes found
Updating TextLCD @ 308d188a2d3a: [Checking]
Mercurial Distributed SCM (version 3.8.4)
(see https://mercurial-scm.org for more information)

Copyright (C) 2005-2016 Matt Mackall and others
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
pulling from https://developer.mbed.org/users/simon/code/TextLCD/
searching for changes
no changes found

```

platformio run

Contents

- *platformio run*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio run [OPTIONS]
pio run [OPTIONS]
```

Description

Process environments which are defined in *Project Configuration File platformio.ini* file

Options

-e, --environment

Process specified environments.

You can also specify which environments should be processed by default using *env_default* option from *Project Configuration File platformio.ini*.

-t, --target

Process specified targets.

Note: You can configure default targets per project environment using *targets* option in *Project Configuration File platformio.ini*.

Built-in targets:

- Processing
 - clean delete compiled object files, libraries and firmware/program binaries
 - upload firmware “auto-uploading” for embedded platforms
 - program firmware “auto-uploading” for embedded platforms using external programmer (available only for *Atmel AVR*)
 - fuses set fuse bits (available only for *Atmel AVR*)
 - buildfs *Uploading files to file system SPIFFS*
 - uploadfs *Uploading files to file system SPIFFS*

- `size` print the size of the sections in a firmware/program
- `checkprogsiz` check maximum allowed firmware size for uploading
- Device
 - `monitor` automatically start *platformio device monitor* after success build operation. You can configure monitor using *Monitor options*.
- Service
 - `envdump` dump current build environment
 - `idedata` export build environment for IDE (defines, build flags, CPPPATH, etc.)

--upload-port

Custom upload port of embedded board. To print all available ports use *platformio device* command.

If upload port is not specified, PlatformIO will try to detect it automatically.

-d, --project-dir

Specify the path to project directory. By default, `--project-dir` is equal to current working directory (CWD).

-s, --silent

Suppress progress reporting

-v, --verbose

Shows detailed information when processing environments.

This option can be set globally using `force_verbose` setting or by environment variable `PLATFORMIO_SETTING_FORCE_VERBOSE`.

--disable-auto-clean

Disable auto-clean of `build_dir` when *Project Configuration File* `platformio.ini` or `src_dir` (project structure) have been modified.

Examples

1. Process Wiring Blink Example

```
> platformio run

[Wed Sep  7 15:48:58 2016] Processing uno (platform: atmelavr, board: uno, framework:arduino)
-----
→-----  

Verbose mode can be enabled via `'-v, --verbose'` option
Collected 36 compatible libraries
Looking for dependencies...
Project does not have dependencies
Compiling .pioenvs/uno/src/main.o
Archiving .pioenvs/uno/libFrameworkArduinoVariant.a
Indexing .pioenvs/uno/libFrameworkArduinoVariant.a
Compiling .pioenvs/uno/FrameworkArduino/CDC.o
...
Compiling .pioenvs/uno/FrameworkArduino/wiring_shift.o
Archiving .pioenvs/uno/libFrameworkArduino.a
Indexing .pioenvs/uno/libFrameworkArduino.a
```

(continues on next page)

(continued from previous page)

```

Linking .pioenvs/uno/firmware.elf
Building .pioenvs/uno/firmware.hex
Calculating size .pioenvs/uno/firmware.elf
AVR Memory Usage
-----
Device: atmega328p

Program:    1034 bytes (3.2% Full)
(.text + .data + .bootloader)

Data:        9 bytes (0.4% Full)
(.data + .bss + .noinit)

=====
 [SUCCESS] Took 2.47 seconds =====

[Wed Sep  7 15:49:01 2016] Processing nodemcu (platform: espressif8266, board:_
→nodemcu, framework: arduino)
-----
→-----
Verbose mode can be enabled via `--v, --verbose` option
Collected 34 compatible libraries
Looking for dependencies...
Project does not have dependencies
Compiling .pioenvs/nodemcu/src/main.o
Archiving .pioenvs/nodemcu/libFrameworkArduinoVariant.a
Indexing .pioenvs/nodemcu/libFrameworkArduinoVariant.a
Compiling .pioenvs/nodemcu/FrameworkArduino/Esp.o
Compiling .pioenvs/nodemcu/FrameworkArduino/FS.o
Compiling .pioenvs/nodemcu/FrameworkArduino/HardwareSerial.o
...
Archiving .pioenvs/nodemcu/libFrameworkArduino.a
Indexing .pioenvs/nodemcu/libFrameworkArduino.a
Linking .pioenvs/nodemcu/firmware.elf
Calculating size .pioenvs/nodemcu/firmware.elf
text      data      bss      dec      hex filename
221240      888    29400   251528   3d688 .pioenvs/nodemcu/firmware.elf
Building .pioenvs/nodemcu/firmware.bin
=====
 [SUCCESS] Took 6.43 seconds =====

[Wed Sep  7 15:49:07 2016] Processing teensy31 (platform: teensy, board: teensy31,_
→framework: arduino)
-----
→-----
Verbose mode can be enabled via `--v, --verbose` option
Collected 96 compatible libraries
Looking for dependencies...
Project does not have dependencies
Compiling .pioenvs/teensy31/src/main.o
Compiling .pioenvs/teensy31/FrameworkArduino/AudioStream.o
Compiling .pioenvs/teensy31/FrameworkArduino/DMAChannel.o
...
Compiling .pioenvs/teensy31/FrameworkArduino/yield.o
Archiving .pioenvs/teensy31/libFrameworkArduino.a
Indexing .pioenvs/teensy31/libFrameworkArduino.a
Linking .pioenvs/teensy31/firmware.elf
Calculating size .pioenvs/teensy31/firmware.elf

```

(continues on next page)

(continued from previous page)

```

text      data      bss      dec      hex filename
11288       168     2288    13744    35b0 .pioenvs/teensy31/firmware.elf
Building .pioenvs/teensy31/firmware.hex
===== [SUCCESS] Took 5.36 seconds =====

[Wed Sep  7 15:49:12 2016] Processing lpmesp430g2553 (platform: timsp430, build_flags:_
→ -D LED_BUILTIN=RED_LED, board: lpmesp430g2553, framework: energia)
-----

→ -----
Verbose mode can be enabled via `'-v, --verbose` option
Collected 29 compatible libraries
Looking for dependencies...
Project does not have dependencies
Compiling .pioenvs/lpmesp430g2553/src/main.o
Compiling .pioenvs/lpmesp430g2553/FrameworkEnergia/HardwareSerial.o
Compiling .pioenvs/lpmesp430g2553/FrameworkEnergia/IPAddress.o
...
Compiling .pioenvs/lpmesp430g2553/FrameworkEnergia/wiring_digital.o
Compiling .pioenvs/lpmesp430g2553/FrameworkEnergia/wiring_pulse.o
Compiling .pioenvs/lpmesp430g2553/FrameworkEnergia/wiring_shift.o
Archiving .pioenvs/lpmesp430g2553/libFrameworkEnergia.a
Indexing .pioenvs/lpmesp430g2553/libFrameworkEnergia.a
Linking .pioenvs/lpmesp430g2553/firmware.elf
Calculating size .pioenvs/lpmesp430g2553/firmware.elf
text      data      bss      dec      hex filename
820        0       20     840     348 .pioenvs/lpmesp430g2553/firmware.elf
Building .pioenvs/lpmesp430g2553/firmware.hex
===== [SUCCESS] Took 2.34 seconds =====

```

2. Process specific environment

```

> platformio run -e nodemcu -e teensy31

[Wed Sep  7 15:49:01 2016] Processing nodemcu (platform: espressif8266, board:_
→ nodemcu, framework: arduino)
-----

→ -----
Verbose mode can be enabled via `'-v, --verbose` option
Collected 34 compatible libraries
Looking for dependencies...
Project does not have dependencies
Compiling .pioenvs/nodemcu/src/main.o
Archiving .pioenvs/nodemcu/libFrameworkArduinoVariant.a
Indexing .pioenvs/nodemcu/libFrameworkArduinoVariant.a
Compiling .pioenvs/nodemcu/FrameworkArduino/Esp.o
Compiling .pioenvs/nodemcu/FrameworkArduino/FS.o
Compiling .pioenvs/nodemcu/FrameworkArduino/HardwareSerial.o
...
Archiving .pioenvs/nodemcu/libFrameworkArduino.a
Indexing .pioenvs/nodemcu/libFrameworkArduino.a
Linking .pioenvs/nodemcu/firmware.elf
Calculating size .pioenvs/nodemcu/firmware.elf
text      data      bss      dec      hex filename
221240     888    29400   251528   3d688 .pioenvs/nodemcu/firmware.elf
Building .pioenvs/nodemcu/firmware.bin
===== [SUCCESS] Took 6.43 seconds =====

```

(continues on next page)

(continued from previous page)

```
[Wed Sep 7 15:49:07 2016] Processing teensy31 (platform: teensy, board: teensy31, framework: arduino)
-----
→-----  

Verbose mode can be enabled via `‐v, --verbose` option  

Collected 96 compatible libraries  

Looking for dependencies...  

Project does not have dependencies  

Compiling .pioenvs/teensy31/src/main.o  

Compiling .pioenvs/teensy31/FrameworkArduino/AudioStream.o  

Compiling .pioenvs/teensy31/FrameworkArduino/DMAChannel.o  

...  

Compiling .pioenvs/teensy31/FrameworkArduino/yield.o  

Archiving .pioenvs/teensy31/libFrameworkArduino.a  

Indexing .pioenvs/teensy31/libFrameworkArduino.a  

Linking .pioenvs/teensy31/firmware.elf  

Calculating size .pioenvs/teensy31/firmware.elf  

text      data      bss      dec      hex filename  

11288       168     2288    13744    35b0 .pioenvs/teensy31/firmware.elf  

Building .pioenvs/teensy31/firmware.hex  

===== [SUCCESS] Took 5.36 seconds =====
```

3. Process specific target (clean project)

```
> platformio run -t clean
[Wed Sep 7 15:53:26 2016] Processing uno (platform: atmelavr, board: uno, framework: arduino)
-----
→-----  

Removed .pioenvs/uno/firmware.elf  

Removed .pioenvs/uno/firmware.hex  

Removed .pioenvs/uno/libFrameworkArduino.a  

Removed .pioenvs/uno/libFrameworkArduinoVariant.a  

Removed .pioenvs/uno/FrameworkArduino/_wiring_pulse.o  

Removed .pioenvs/uno/FrameworkArduino/abi.o  

Removed .pioenvs/uno/FrameworkArduino/CDC.o  

Removed .pioenvs/uno/FrameworkArduino/HardwareSerial.o  

Removed .pioenvs/uno/FrameworkArduino/HardwareSerial0.o  

Removed .pioenvs/uno/FrameworkArduino/HardwareSerial1.o  

Removed .pioenvs/uno/FrameworkArduino/HardwareSerial2.o  

Removed .pioenvs/uno/FrameworkArduino/HardwareSerial3.o  

Removed .pioenvs/uno/FrameworkArduino/hooks.o  

Removed .pioenvs/uno/FrameworkArduino/IPAddress.o  

Removed .pioenvs/uno/FrameworkArduino/main.o  

Removed .pioenvs/uno/FrameworkArduino/new.o  

Removed .pioenvs/uno/FrameworkArduino/PluggableUSB.o  

Removed .pioenvs/uno/FrameworkArduino/Print.o  

Removed .pioenvs/uno/FrameworkArduino/Stream.o  

Removed .pioenvs/uno/FrameworkArduino/Tone.o  

Removed .pioenvs/uno/FrameworkArduino/USBCore.o  

Removed .pioenvs/uno/FrameworkArduino/WIInterrupts.o  

Removed .pioenvs/uno/FrameworkArduino/wiring.o  

Removed .pioenvs/uno/FrameworkArduino/wiring_analog.o  

Removed .pioenvs/uno/FrameworkArduino/wiring_digital.o  

Removed .pioenvs/uno/FrameworkArduino/wiring_pulse.o  

Removed .pioenvs/uno/FrameworkArduino/wiring_shift.o  

Removed .pioenvs/uno/FrameworkArduino/WMath.o
```

(continues on next page)

(continued from previous page)

```

Removed .pioenvs/uno/FrameworkArduino/WString.o
Removed .pioenvs/uno/src/main.o
Done cleaning
===== [SUCCESS] Took 0.49 seconds =====

[Wed Sep 7 15:53:27 2016] Processing nodemcu (platform: espressif8266, board:_
↳ nodemcu, framework: arduino)

=====
Removed .pioenvs/nodemcu/firmware.bin
Removed .pioenvs/nodemcu/firmware.elf
Removed .pioenvs/nodemcu/libFrameworkArduino.a
Removed .pioenvs/nodemcu/libFrameworkArduinoVariant.a
...
Removed .pioenvs/nodemcu/FrameworkArduino/spiffs/spiffs_nucleus.o
Removed .pioenvs/nodemcu/FrameworkArduino/umm_malloc/umm_malloc.o
Removed .pioenvs/nodemcu/src/main.o
Done cleaning
===== [SUCCESS] Took 0.50 seconds =====

[Wed Sep 7 15:53:27 2016] Processing teensy31 (platform: teensy, board: teensy31,_
↳ framework: arduino)

=====
Removed .pioenvs/teensy31/firmware.elf
Removed .pioenvs/teensy31/firmware.hex
Removed .pioenvs/teensy31/libFrameworkArduino.a
Removed .pioenvs/teensy31/FrameworkArduino/analog.o
Removed .pioenvs/teensy31/FrameworkArduino/AudioStream.o
...
Removed .pioenvs/teensy31/FrameworkArduino/WString.o
Removed .pioenvs/teensy31/FrameworkArduino/yield.o
Removed .pioenvs/teensy31/src/main.o
Done cleaning
===== [SUCCESS] Took 0.50 seconds =====

[Wed Sep 7 15:53:28 2016] Processing lpmsp430g2553 (platform: tmsp430, build_flags:_
↳ -D LED_BUILTIN=RED_LED, board: lpmsp430g2553, framework: energia)

=====
Removed .pioenvs/lpmsp430g2553/firmware.elf
Removed .pioenvs/lpmsp430g2553/firmware.hex
Removed .pioenvs/lpmsp430g2553/libFrameworkEnergia.a
Removed .pioenvs/lpmsp430g2553/FrameworkEnergia/atof.o
...
Removed .pioenvs/lpmsp430g2553/FrameworkEnergia/avr/dtostrf.o
Removed .pioenvs/lpmsp430g2553/src/main.o
Done cleaning
===== [SUCCESS] Took 0.49 seconds =====

```

4. Mix environments and targets

```

> platformio run -e uno -t upload

[Wed Sep 7 15:55:11 2016] Processing uno (platform: atmelavr, board: uno, framework:_
↳ arduino)
=====

(continues on next page)

```

(continued from previous page)

```
Verbose mode can be enabled via `--verbose` option
Collected 36 compatible libraries
Looking for dependencies...
Project does not have dependencies
Compiling .pioenvs/uno/src/main.o
Archiving .pioenvs/uno/libFrameworkArduinoVariant.a
Indexing .pioenvs/uno/libFrameworkArduinoVariant.a
Compiling .pioenvs/uno/FrameworkArduino/CDC.o
...
Compiling .pioenvs/uno/FrameworkArduino/wiring_shift.o
Archiving .pioenvs/uno/libFrameworkArduino.a
Indexing .pioenvs/uno/libFrameworkArduino.a
Linking .pioenvs/uno/firmware.elf
Checking program size .pioenvs/uno/firmware.elf
text      data      bss      dec      hex filename
1034        0        9    1043     413 .pioenvs/uno/firmware.elf
Building .pioenvs/uno/firmware.hex
Looking for upload port...
Auto-detected: /dev/cu.usbmodemFA141
Uploading .pioenvs/uno/firmware.hex

avrduude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.01s

avrduude: Device signature = 0x1e950f
avrduude: reading input file ".pioenvs/uno/firmware.hex"
avrduude: writing flash (1034 bytes):

Writing | ##### | 100% 0.18s

avrduude: 1034 bytes of flash written
avrduude: verifying flash memory against .pioenvs/uno/firmware.hex:
avrduude: load data flash data from input file .pioenvs/uno/firmware.hex:
avrduude: input file .pioenvs/uno/firmware.hex contains 1034 bytes
avrduude: reading on-chip flash data:

Reading | ##### | 100% 0.15s

avrduude: verifying ...
avrduude: 1034 bytes of flash verified

avrduude: safemode: Fuses OK (H:00, E:00, L:00)

avrduude done. Thank you.

===== [SUCCESS] Took 4.14 seconds =====
```

platformio settings

Manage PlatformIO settings

Contents

- *platformio settings*
 - *platformio settings get*
 - * *Usage*
 - * *Description*
 - * *Settings*
 - *auto_update_libraries*
 - *auto_update_platforms*
 - *check_libraries_interval*
 - *check_platformio_interval*
 - *check_platforms_interval*
 - *enable_cache*
 - *enable_ssl*
 - *enable_telemetry*
 - *force_verbose*
 - *projects_dir*
 - * *Examples*
 - *platformio settings set*
 - * *Usage*
 - * *Description*
 - * *Examples*
 - *platformio settings reset*
 - * *Usage*
 - * *Description*
 - * *Examples*

platformio settings get

Usage

```
platformio settings get [NAME]
pio settings get [NAME]
```

Description

Note:

- The `Yes` value is equal to: `True`, `Y`, `1` and is not case sensitive.
 - You can override these settings using *Environment variables*.
-

Get/List existing settings

Settings

`auto_update_libraries`

Default No

Values Yes/No

Automatically update libraries.

`auto_update_platforms`

Default No

Values Yes/No

Automatically update platforms.

`check_libraries_interval`

Default 7

Values Days (Number)

Check for the library updates interval.

`check_platformio_interval`

Default 3

Values Days (Number)

Check for the new PlatformIO interval.

`check_platforms_interval`

Default 7

Values Days (Number)

Check for the platform updates interval.

enable_cache**Default** Yes**Values** Yes/No

Enable caching for API requests and Library Manager

enable_ssl**Default** No**Values** Yes/No

Enable SSL for PlatformIO Services

enable_telemetry**Default** Yes**Values** Yes/No

Share diagnostics and usage information to help us make PlatformIO better:

- PlatformIO errors/exceptions
- The name of used platforms, boards, frameworks (for example, “espressif”, “arduino”, “uno”, etc.)
- The name of commands (for example, “run”, “lib list”, etc.)
- The type of IDE (for example, “atom”, “eclipse”, etc.)

This gives us a sense of what parts of the PlatformIO is most important.

The source code of telemetry service is [open source](#). You can make sure that we DO NOT share PRIVATE information or source code of your project. All information shares anonymously.

Thanks a lot that keep this setting enabled.

force_verbose**Default** No**Values** Yes/No

Force verbose output when processing environments. This setting overrides

- `platformio run --verbose`
- `platformio ci --verbose`
- `platformio test --verbose`

projects_dir**Default** ~/Documents/PlatformIO/Projects**Values** Path to folder

Default location for PlatformIO projects (PIO Home)

Examples

1. List all settings and theirs current values

```
> platformio settings get
```

Name	Value [Default]	Description
auto_update_libraries	No	Automatically update libraries (Yes/ No)
auto_update_platforms	No	Automatically update platforms (Yes/ No)
check_libraries_interval	7	Check for the library updates interval (days)
check_platformio_interval	3	Check for the new PlatformIO interval (days)
check_platforms_interval	7	Check for the platform updates interval (days)
enable_cache	Yes	Enable caching for API requests and Library Manager
enable_ssl	No	Enable SSL for PlatformIO Services
enable_telemetry	Yes	Telemetry service?#enable-telemetry> (Yes/No)
force_verbose	No	Force verbose output when processing environments
projects_dir	~/Documents/PlatformIO/Projects	Default location for PlatformIO projects (PIO Home)

2. Show specified setting

```
$ platformio settings get auto_update_platforms
```

Name	Value [Default]	Description
auto_update_platforms	Yes	Automatically update platforms (Yes/ No)

platformio settings set

Usage

```
platformio settings set NAME VALUE
```

Description

Set new value for the setting

Examples

Change to check for the new PlatformIO each day

```
$ platformio settings set check_platformio_interval 1
The new value for the setting has been set!
Name           Value [Default]   Description
-----
check_platformio_interval    1 [3]      Check for the new PlatformIO_
interval (days)
```

platformio settings reset

Usage

```
platformio settings reset
```

Description

Reset settings to default

Examples

```
$ platformio settings reset
The settings have been reseted!

Name           Value [Default]   Description
-----
auto_update_libraries    No        Automatically update libraries (Yes/
No)
auto_update_platforms    No        Automatically update platforms (Yes/
No)
check_libraries_interval 7        Check for the library updates_
interval (days)
check_platformio_interval 3        Check for the new PlatformIO_
interval (days)
check_platforms_interval 7        Check for the platform updates_
interval (days)
enable_cache            Yes       Enable caching for API requests and_
Library Manager
enable_ssl              No        Enable SSL for PlatformIO Services
enable_telemetry         Yes      Telemetry service?#enable-telemetry>
(Yes/No)
force_verbose           No        Force verbose output when_
processing environments
projects_dir             ~/Documents/PlatformIO/Projects Default location for_#
PlatformIO projects (PIO Home)
```

platformio test

Helper command for local *PIO Unit Testing*.

Contents

- *platformio test*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio test [OPTIONS]
pio test [OPTIONS]
```

Description

Run locally tests from PlatformIO based project. More details about PlatformIO [PIO Unit Testing](#).

This command allows you to apply the tests for the environments specified in [Project Configuration File platformio.ini](#).

Options

-e, --environment

Process specified environments. More details [platformio run --environment](#)

-f, --filter

Process only the tests where the name matches specified patterns. More than one pattern is allowed. If you need to filter some tests for a specific environment, please take a look at *test_filter* option from [Project Configuration File platformio.ini](#).

Pattern	Meaning
*	matches everything
?	matches any single character
[seq]	matches any character in seq
[!seq]	matches any character not in seq

For example, `platformio test --filter "mytest*" -i "test[13]"`

-i, --ignore

Ignore tests where the name matches specified patterns. More than one pattern is allowed. If you need to ignore some tests for a specific environment, please take a look at *test_ignore* option from [Project Configuration File platformio.ini](#).

Pattern	Meaning
*	matches everything
?	matches any single character
[seq]	matches any character in seq
[!seq]	matches any character not in seq

For example, `platformio test --ignore "mytest*" -i "test[13]"`

--upload-port

A port that is intended for firmware uploading. To list available ports please use `platformio device list` command.

If upload port is not specified, PlatformIO will try to detect it automatically.

--test-port

A Serial/UART port that PlatformIO uses as communication interface between PlatformIO Unit Test Engine and target device. To list available ports please use `platformio device list` command.

If test port is not specified, PlatformIO will try to detect it automatically.

-d, --project-dir

Specify the path to project directory. By default, `--project-dir` is equal to current working directory (CWD).

--without-building

Skip building stage.

--without-uploading

Skip uploading stage

--no-reset

Disable software reset via `Serial.DTR/RST` before test running. In this case, need to press “reset” button manually after firmware uploading.

Warning: If board does not support software reset via `Serial.DTR/RTS` you should add >2 seconds delay before `UNITY_BEGIN()```. We need that time to establish a ``Serial communication between host machine and target device. See [PIO Unit Testing](#).

--monitor-rts

Set initial RTS line state for Serial Monitor (0 or 1), default 1. We use it to gather test results via Serial connection.

--monitor-dtr

Set initial DTR line state for Serial Monitor (0 or 1), default 1. We use it to gather test results via Serial connection.

-v, --verbose

Shows detailed information when processing environments.

This option can be set globally using `force_verbose` setting or by environment variable `PLATFORMIO_SETTING_FORCE_VERBOSE`.

Examples

For the examples please follow to [PIO Unit Testing](#) page.

platformio update

Contents

- *platformio update*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio update [OPTIONS]
pio update [OPTIONS]
```

Description

Check or update installed PIO Core packages, *Development Platforms* and global *Libraries*. This command is combination of 2 sub-commands:

- *platformio platform update*
- *platformio lib update*

Options

--core-packages

Update only the core packages

-c, --only-check

Do not update, only check for new version

Examples

```
> platformio update

Platform Manager
=====
Platform timsp430
-----
Updating timsp430 @ 0.0.0: [Up-to-date]
Updating toolchain-timsp430 @ 1.40603.0: [Up-to-date]
Updating framework-energiamsp430 @ 1.17.0: [Up-to-date]
Updating framework-arduinomsp430 @ 1.10601.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform freescalekinetis
-----
Updating freescalekinetis @ 0.0.0: [Up-to-date]
```

(continues on next page)

(continued from previous page)

```

Updating framework-mbed @ 1.121.1: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform ststm32
-----
Updating ststm32 @ 0.0.0: [Up-to-date]
Updating framework-libopencm3 @ 1.1.0: [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0: [Up-to-date]
Updating tool-stlink @ 1.10200.0: [Up-to-date]
Updating framework-spl @ 1.10201.0: [Up-to-date]
Updating framework-cmsis @ 1.40300.0: [Up-to-date]
Updating framework-mbed @ 1.121.1: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform lattice_ice40
-----
Updating lattice_ice40 @ 0.0.0: [Up-to-date]
Updating toolchain-icestorm @ 1.7.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform atmelavr
-----
Updating atmelavr @ 0.0.0: [Up-to-date]
Updating framework-arduinoavr @ 1.10608.1: [Up-to-date]
Updating tool-avrdude @ 1.60001.1: [Up-to-date]
Updating toolchain-atmelavr @ 1.40801.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform espressif8266
-----
Updating espressif8266 @ 0.0.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]
Updating toolchain-xtensa @ 1.40802.0: [Up-to-date]
Updating tool-esptool @ 1.409.0: [Up-to-date]
Updating tool-mkspiffs @ 1.102.0: [Up-to-date]
Updating framework-arduinonespressif8266 @ 1.20300.0: [Up-to-date]
Updating sdk-esp8266 @ 1.10502.0: [Up-to-date]

Platform linux_x86_64
-----
Updating linux_x86_64 @ 0.0.0: [Up-to-date]
Updating toolchain-gcclinux64 @ 1.40801.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform windows_x86
-----
Updating windows_x86 @ 0.0.0: [Up-to-date]
Updating toolchain-gccmingw32 @ 1.40800.0: [Up-to-date]
Updating tool-scons @ 2.4.1: [Up-to-date]

Platform teensy
-----
Updating teensy @ 0.0.0: [Up-to-date]
Updating framework-arduinoteensy @ 1.128.0: [Up-to-date]
Updating tool-teensy @ 1.1.0: [Up-to-date]
Updating framework-mbed @ 1.121.1: [Up-to-date]

```

(continues on next page)

(continued from previous page)

```

Updating tool-scons @ 2.4.1:      [Up-to-date]
Updating toolchain-atmelavr @ 1.40801.0:      [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0:      [Up-to-date]

Platform nordicnrf51
-----
Updating nordicnrf51 @ 0.0.0:      [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0:      [Up-to-date]
Updating framework-arduino nordicnrf51 @ 1.20302.0:      [Up-to-date]
Updating framework-mbed @ 1.121.1:      [Up-to-date]
Updating tool-scons @ 2.4.1:      [Up-to-date]

Platform titiva
-----
Updating titiva @ 0.0.0:      [Up-to-date]
Updating framework-libopencm3 @ 1.1.0:      [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0:      [Up-to-date]
Updating framework-energiativa @ 1.17.0:      [Up-to-date]
Updating tool-scons @ 2.4.1:      [Up-to-date]

Platform atmelsam
-----
Updating atmelsam @ 0.0.0:      [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0:      [Up-to-date]
Updating tool-openocd @ 1.900.0:      [Up-to-date]
Updating framework-mbed @ 1.121.1:      [Up-to-date]
Updating tool-scons @ 2.4.1:      [Up-to-date]
Updating tool-avrdude @ 1.60001.1:      [Up-to-date]
Updating tool-bossac @ 1.10601.0:      [Up-to-date]

Platform siliconlabsefm32
-----
Updating siliconlabsefm32 @ 0.0.0:      [Up-to-date]
Updating framework-mbed @ 1.121.1:      [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0:      [Up-to-date]
Updating tool-scons @ 2.4.1:      [Up-to-date]

Platform microchippic32
-----
Updating microchippic32 @ 0.0.0:      [Up-to-date]
Updating framework-arduino microchippic32 @ 1.10201.0:      [Up-to-date]
Updating toolchain-microchippic32 @ 1.40803.0:      [Up-to-date]
Updating tool-pic32prog @ 1.200200.0:      [Up-to-date]
Updating tool-scons @ 2.4.1:      [Up-to-date]

Platform linux_i686
-----
Updating linux_i686 @ 0.0.0:      [Up-to-date]
Updating toolchain-gcclinix32 @ 1.40801.0:      [Up-to-date]
Updating tool-scons @ 2.4.1:      [Up-to-date]

Platform intel_arc32
-----
Updating intel_arc32 @ 0.0.0:      [Up-to-date]
Updating framework-arduino intel @ 1.10006.0:      [Up-to-date]
Updating tool-arduino101load @ 1.124.0:      [Up-to-date]
Updating toolchain-intelarc32 @ 1.40805.0:      [Up-to-date]

```

(continues on next page)

(continued from previous page)

```

Updating tool-scons @ 2.4.1:      [Up-to-date]

Platform nxplpc
-----
Updating nxplpc @ 0.0.0:      [Up-to-date]
Updating framework-mbed @ 1.121.1:  [Up-to-date]
Updating toolchain-gccarmnoneabi @ 1.40804.0:  [Up-to-date]
Updating tool-scons @ 2.4.1:      [Up-to-date]

Platform linux_arm
-----
Updating linux_arm @ 0.0.0:      [Up-to-date]
Updating toolchain-gccarmlinuxgnueabi @ 1.40802.0:  [Up-to-date]
Updating tool-scons @ 2.4.1:      [Up-to-date]

Platform native
-----
Updating native @ 0.0.0:      [Up-to-date]
Updating tool-scons @ 2.4.1:      [Up-to-date]

Library Manager
=====
Updating Adafruit-GFX @ 334e815bc1:      [Up-to-date]
Updating Adafruit-ST7735 @ d53d4bf03a:  [Up-to-date]
Updating Adafruit-DHT @ 09344416d2:      [Up-to-date]
Updating Adafruit-Unified-Sensor @ f2af6f4efc:  [Up-to-date]
Updating ESP8266_SSD1306 @ 3.2.3:      [Up-to-date]
Updating EngduinoMagnetometer @ 3.1.0:  [Up-to-date]
Updating IRremote @ 2.2.1:      [Up-to-date]
Updating Json @ 5.6.4:      [Up-to-date]
Updating MODSERIAL @ d8422efe47:      [Up-to-date]
Updating PJSON @ 1fb26fd:      [Checking]
git version 2.7.4 (Apple Git-66)
Already up-to-date.
Updating Servo @ 36b69a7ced07:  [Checking]
Mercurial Distributed SCM (version 3.8.4)
(see https://mercurial-scm.org for more information)

Copyright (C) 2005-2016 Matt Mackall and others
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
pulling from https://developer.mbed.org/users/simon/code/Servo/
searching for changes
no changes found
Updating TextLCD @ 308d188a2d3a:      [Checking]
Mercurial Distributed SCM (version 3.8.4)
(see https://mercurial-scm.org for more information)

Copyright (C) 2005-2016 Matt Mackall and others
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
pulling from https://developer.mbed.org/users/simon/code/TextLCD/
searching for changes
no changes found

```

platformio upgrade

Contents

- *platformio upgrade*
 - *Usage*
 - *Description*
 - *Options*
 - *Examples*

Usage

```
platformio upgrade
pio upgrade
```

Description

Check or upgrade PlatformIO to the latest version

Options

--dev

Use development branch.

Examples

```
> platformio upgrade
You are up-to-date!
PlatformIO x.x.x is currently the newest version available.

# If you have problem with permissions try:
> sudo platformio upgrade
```

1.4 PlatformIO Home

PIO Home allows you to interact with PlatformIO ecosystem using modern and cross-platform GUI:

- PlatformIO Projects
- *PIO Account*
- *Library Manager*
- *Development Platforms*

- *Frameworks*
- *Embedded Boards*
- *Device Manager*: serial, logical, and multicast DNS services
- Library and development platform updates.

1.4.1 Installation

You do not need to install **PIO Home** separately, it's already built-in in *PlatformIO IDE* and *PlatformIO Core*.

1.4.2 Quick Start

PlatformIO IDE

Please open **PIO Home** using (HOME) button on PIO Toolbar:

- **Atom**: *PlatformIO Toolbar*
- **VSCode**: *PlatformIO Toolbar*

PlatformIO Core

Please launch **PIO Home** Web-server using *platformio home* command and open in your browser <http://127.0.0.1:8008>.

You can change host and port. Please check *platformio home* command for details.

1.4.3 Demo

Welcome & Project Manager

The screenshot shows the PlatformIO Home interface. On the left is a dark sidebar with icons for Home, Account, Libraries (with 3 notifications), Boards, Platforms, and Devices. The main area features a large orange logo of a bug-like character with two antennae and two large eyes. To the right of the logo is a "Quick Access" section with four buttons: "+ New Project", "Import Arduino Project", "Open Project", and "Project Examples". Below the logo, it says "Home 0.6.0 · Core 3.5.0". At the bottom of the main area, there's a navigation bar with links to Web, Open Source, Get Started, Docs, News, Community, Report an Issue, Donate, and Contact.

Recent Projects

Search project...

Name	Boards	Modified	Action
debug/vscode	Arduino M0 Pro (Programming/Debug Port)	14 hours ago	Hide Open
unit-testing/calculator	Arduino Uno , NodeMCU 1.0 (ESP-12E Module)	18 hours ago	Hide Open
examples/wiring-blink	Arduino Uno , NodeMCU 0.9 (ESP-12 Module) , Teensy 3.1 / 3.2 , TI LaunchPad MSP- EXP430G2553LP	23 hours ago	Hide Open
debug/mbed	ST Nucleo L053RB	12 days ago	Hide Open

If you enjoy using PlatformIO, please star our projects on GitHub!

Library Manager

The screenshot shows the PlatformIO Library Manager interface. On the left is a sidebar with icons for Home, Account, Libraries (selected), Boards, Platforms, and Devices. The main area has tabs for Registry (selected), Installed, Built-in, and Updates (with 3 notifications). A search bar at the top right includes filters for tft display, header:RH_ASK.h, keyword:mqtt, framework:mbed, and more... A "Have an account? Log in" link is also present.

Recently

Buttons for All Libraries, Register, and Advanced are available.

Updated	Added	Keywords
U8g2 3 hours ago	esp8266ndn 3 hours ago	sha1
esp8266ndn 3 hours ago	Cryptosuite 1 day ago	hmac
Adafruit BME280 Library 12 hours ago	CryptoC 1 day ago	upnp
IHCSoapClient 17 hours ago	Crypto 1 day ago	smartthings
RapidJSON 20 hours ago	NTptimeESP 1 day ago	ssdp

Popular Tags

A grid of popular tags:

- display, communication, sensors, control
- device, lcd, graphics, tft, oled
- displaycore, glcd, other, input, signal
- output, data, font, sensor, i2c, spi
- timing, esp8266, processing, storage, serial
- wifi, temperature, http, arduino, iot, rf, led, radio
- web, i2cdevlib, ethernet, uncategorized, time, timer
- wireless, mqtt, mbed, server, protocol, accelerometer
- wi-fi, button, rtc, humidity, rest

Trending

1.4. PlatformIO Home

Today	Week	Month
▲ SerialESP8266wifi	▲ PubSubClient	▲ ArduinoJson

125

Board Explorer

The screenshot shows the PlatformIO Board Explorer interface. On the left is a dark sidebar with icons for Home, Account, Libraries (with 3 notifications), Boards (selected), Platforms, and Devices. The main area has a header with navigation buttons and a 'Have an account? Log in' link. Below is a section titled 'Board Explorer' with a count of 469 boards. A message states: 'PlatformIO currently supports over 400 boards from leading manufacturers, and we are constantly adding new ones. You can be part of the process by letting us know what board you wish to see supported next, by [submitting a feature request](#)'. A search bar and filter buttons for 'Clear filters', 'IoT-enabled', and 'Debugger' are present. The main table lists boards with columns for Name, Platform, Frameworks, MCU, FRQ, ROM, RAM, and Extra. The first few rows are:

Name	Platform	Frameworks	MCU	FRQ	ROM	RAM	Extra	
1Bitisy	ST STM32	CMSIS, libOpenCM3, SPL, STM32Cube	STM32F415RG	168 Mhz	1 MB	128 KB		
4DSystems PICadillo 35T	Microchip PIC32	Arduino	32MX7L	90F512	80 Mhz	508 KB	128 KB	
96Boards B96B-F446VE	ST STM32	mbed, STM32Cube	STM32F446VET6	168 Mhz	512 KB	128 KB		
Adafruit Bluefruit Micro	Atmel AVR	Arduino	ATMEGA32U4	8 Mhz	28 KB	2.5 KB		
Adafruit Circuit Playground Express	Atmel SAM	Arduino	SAMD21G18A	48 Mhz	256 KB	32 KB		
Adafruit ESP32 Feather	Espressif 32	Arduino, ESP-IDF	ESP32	240 Mhz	1 MB	288 KB		

1.5 Tutorials and Examples

1.5.1 Tutorials

Unit Testing of a “Blink” Project

The goal of this tutorial is to demonstrate how is simple to use [PIO Unit Testing](#).

- **Level:** Beginner
- **Platforms:** Windows, macOS, Linux

Contents

- [Setting Up the Project](#)
- [Project structure](#)
- [Source files](#)
- [Test results](#)

Setting Up the Project

1. Please navigate to [Quick Start](#) section and create the “Blink Project”.
2. Create a `test` directory in the project (on the same level as `src`) and place `test_main.cpp` file in it (the source code is located below).
3. Run tests using `platformio test` command.

Project structure

```
project_dir
├── lib
│   └── readme.txt
├── platformio.ini
└── src
    └── ...
└── test
    └── test_main.cpp
```

Source files

- `platformio.ini`

```
; PlatformIO Project Configuration File
;
; Build options: build flags, source filter, extra scripting
; Upload options: custom port, speed and extra flags
; Library options: dependencies, extra library storages
;
```

(continues on next page)

(continued from previous page)

```
; Please visit documentation for the other options and examples
; http://docs.platformio.org/page/projectconf.html
```

```
[env:uno]
platform = atmelavr
framework = arduino
board = uno

[env:nodemcu]
platform = espressif8266
framework = arduino
board = nodemcu

[env:teensy31]
platform = teensy
framework = arduino
board = teensy31
```

- test/test_main.cpp

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

// void setUp(void) {
// // set stuff up here
// }

// void tearDown(void) {
// // clean stuff up here
// }

void test_led_builtin_pin_number(void) {
    TEST_ASSERT_EQUAL(LED_BUILTIN, 13);
}

void test_led_state_high(void) {
    digitalWrite(LED_BUILTIN, HIGH);
    TEST_ASSERT_EQUAL(digitalRead(LED_BUILTIN), HIGH);
}

void test_led_state_low(void) {
    digitalWrite(LED_BUILTIN, LOW);
    TEST_ASSERT_EQUAL(digitalRead(LED_BUILTIN), LOW);
}

void setup() {
    // NOTE!!! Wait for >2 secs
    // if board doesn't support software reset via Serial.DTR/RTS
    delay(2000);

    UNITY_BEGIN();      // IMPORTANT LINE!
    RUN_TEST(test_led_builtin_pin_number);

    pinMode(LED_BUILTIN, OUTPUT);
}
```

(continues on next page)

(continued from previous page)

```

uint8_t i = 0;
uint8_t max_blinks = 5;

void loop() {
    if (i < max_blinks)
    {
        RUN_TEST(test_led_state_high);
        delay(500);
        RUN_TEST(test_led_state_low);
        delay(500);
        i++;
    }
    else if (i == max_blinks) {
        UNITY_END(); // stop unit testing
    }
}

```

Test results

```

> platformio test -e nodemcu --verbose

PIO Plus (https://pioplus.com) v0.1.0
Verbose mode can be enabled via `--v, --verbose` option
Collected 1 items

===== [test/*] Building... (1/3) ↴
=====
[Wed Sep 7 15:16:55 2016] Processing nodemcu (platform: espressif8266, board: nodemcu, framework: arduino)
-----
=====
Verbose mode can be enabled via `--v, --verbose` option
Collected 34 compatible libraries
Looking for dependencies...
Project does not have dependencies
Compiling .pioenvs/nodemcu/src/main.o
Compiling .pioenvs/nodemcu/test/output_export.o
Compiling .pioenvs/nodemcu/test/test_main.o
Compiling .pioenvs/nodemcu/UnityTestLib/unity.o
Archiving .pioenvs/nodemcu/libFrameworkArduinoVariant.a
Indexing .pioenvs/nodemcu/libFrameworkArduinoVariant.a
Compiling .pioenvs/nodemcu/FrameworkArduino/Esp.o
Compiling .pioenvs/nodemcu/FrameworkArduino/FS.o
Compiling .pioenvs/nodemcu/FrameworkArduino/HardwareSerial.o
Compiling .pioenvs/nodemcu/FrameworkArduino/IPAddress.o
Archiving .pioenvs/nodemcu/libUnityTestLib.a
Indexing .pioenvs/nodemcu/libUnityTestLib.a
Compiling .pioenvs/nodemcu/FrameworkArduino/MD5Builder.o
...
Compiling .pioenvs/nodemcu/FrameworkArduino/umm_malloc/umm_malloc.o
Archiving .pioenvs/nodemcu/libFrameworkArduino.a
Indexing .pioenvs/nodemcu/libFrameworkArduino.a
Linking .pioenvs/nodemcu/firmware.elf
Calculating size .pioenvs/nodemcu/firmware.elf

```

(continues on next page)

(continued from previous page)

```

text      data      bss      dec      hex filename
223500    2408    29536  255444   3e5d4 .pioenvs/nodemcu/firmware.elf
Building .pioenvs/nodemcu/firmware.bin

===== [test/*] Uploading... (2/3) ↴
=====
[Wed Sep  7 15:17:01 2016] Processing nodemcu (platform: espressif8266, board: ↴
↳ nodemcu, framework: arduino)
-----

↳
Verbose mode can be enabled via `--verbose` option
Collected 34 compatible libraries
Looking for dependencies...
Project does not have dependencies
Linking .pioenvs/nodemcu/firmware.elf
Checking program size .pioenvs/nodemcu/firmware.elf
text      data      bss      dec      hex filename
223500    2408    29536  255444   3e5d4 .pioenvs/nodemcu/firmware.elf
Calculating size .pioenvs/nodemcu/firmware.elf
text      data      bss      dec      hex filename
223500    2408    29536  255444   3e5d4 .pioenvs/nodemcu/firmware.elf
Looking for upload port...
Auto-detected: /dev/cu.SLAB_USBtoUART
Uploading .pioenvs/nodemcu/firmware.bin
Uploading 230064 bytes from .pioenvs/nodemcu/firmware.bin to flash at 0x00000000
..... [ 35
↳ % ]
..... [ 71
↳ % ]
..... [ 100
↳ % ]

===== [test/*] Testing... (3/3) ↴
=====
If you don't see any output for the first 10 secs, please reset board (press reset ↴
button)

test/test_main.cpp:41:test_led_state_high      [PASSED]
test/test_main.cpp:43:test_led_state_low       [PASSED]
test/test_main.cpp:41:test_led_state_high      [PASSED]
test/test_main.cpp:43:test_led_state_low       [PASSED]
test/test_main.cpp:41:test_led_state_high      [PASSED]
test/test_main.cpp:43:test_led_state_low       [PASSED]
test/test_main.cpp:41:test_led_state_high      [PASSED]
test/test_main.cpp:43:test_led_state_low       [PASSED]
-----
11 Tests 1 Failures 0 Ignored

===== [TEST SUMMARY] ↴
=====
test/*/env:nodemcu      [PASSED]
===== [PASSED] Took 38.15 seconds ↴
=====
```

STM32Cube HAL and Nucleo-F401RE: debugging and unit testing

The goal of this tutorial is to demonstrate how simple it is to use [PlatformIO IDE for Atom](#) to develop, run and debug a basic blink project with [STM32Cube](#) framework for STM32 Nucleo-F401RE board.

- **Level:** Intermediate
- **Platforms:** Windows, Mac OS X, Linux

Requirements:

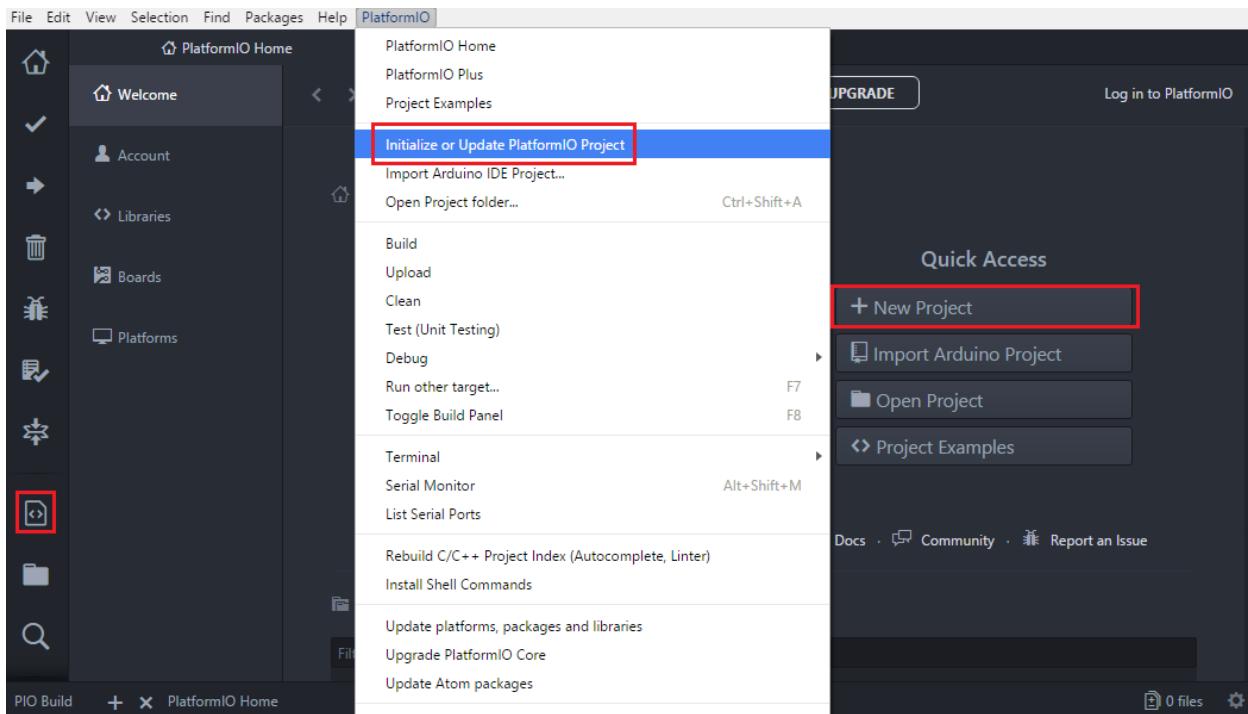
- Downloaded and installed [PlatformIO IDE for Atom](#)
- Nucleo-F401RE development board

Contents

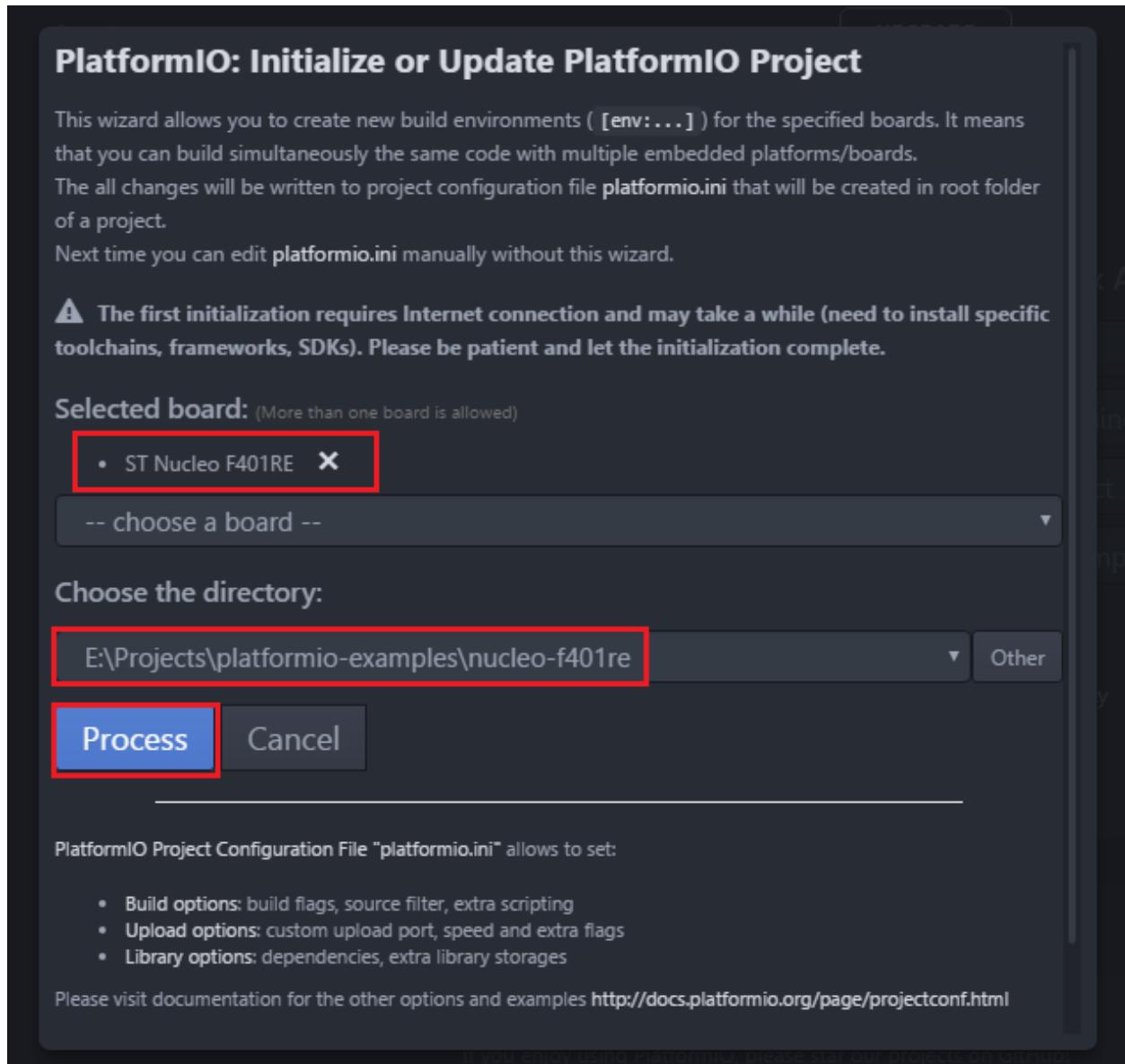
- [*Setting Up the Project*](#)
- [*Adding Code to the Generated Project*](#)
- [*Compiling and Uploading the Firmware*](#)
- [*Debugging the Firmware*](#)
- [*Writing Unit Tests*](#)
- [*Conclusion*](#)
- [*Project Source Code*](#)

Setting Up the Project

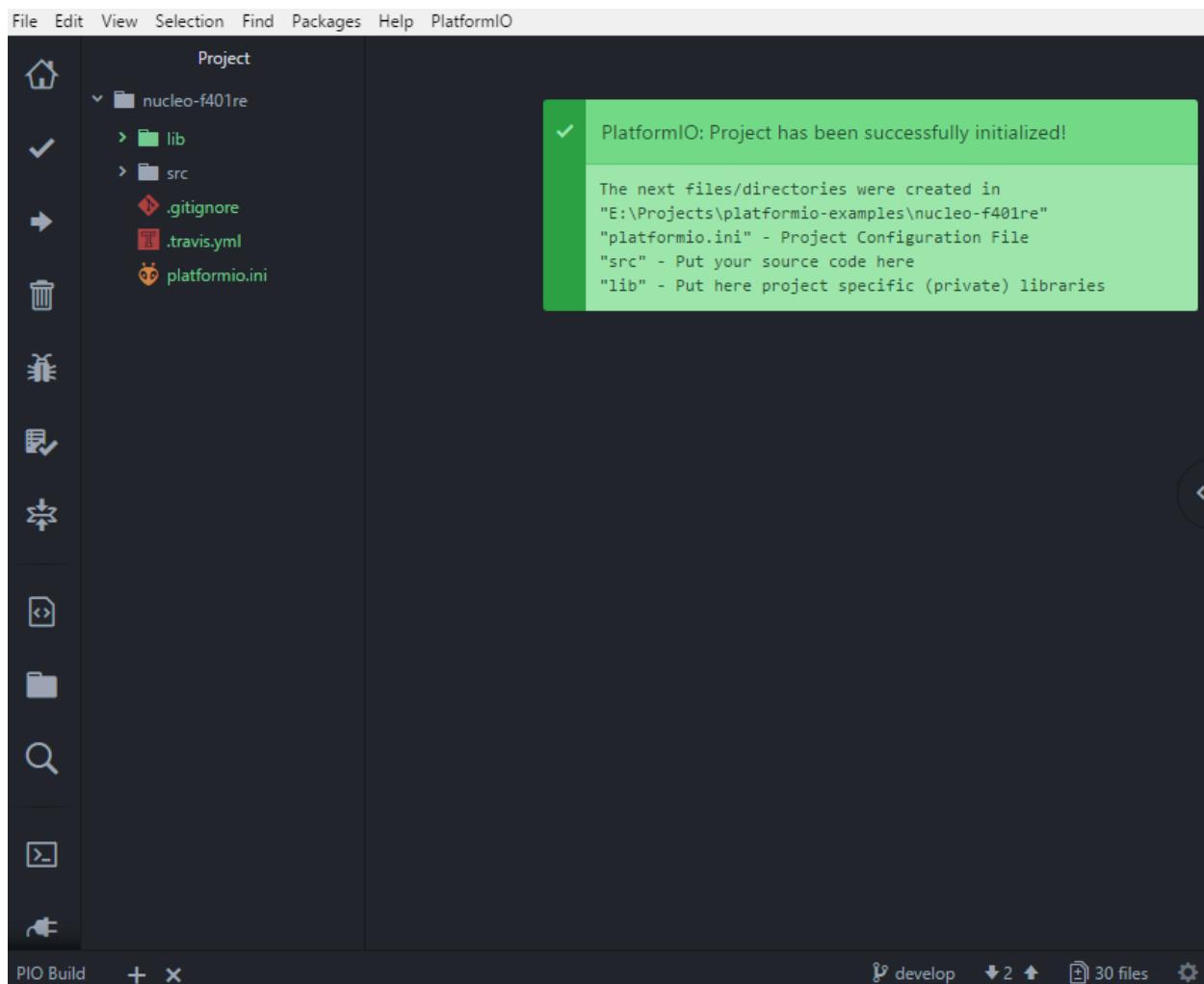
There are two ways how to create a new project in PlatformIO IDE: using “New Project” button menu on Home Page or using Menu: PlatformIO > Initialize or Update PlatformIO Project:



On the next step we choose the desired board (in our case it's ST Nucleo-F401RE) and also select a directory for our project:



After processing the selected options (PlatformIO IDE will download and install all required packages, thus the first installation may take some amount of time), we should now have the new project created with the following folder structure:



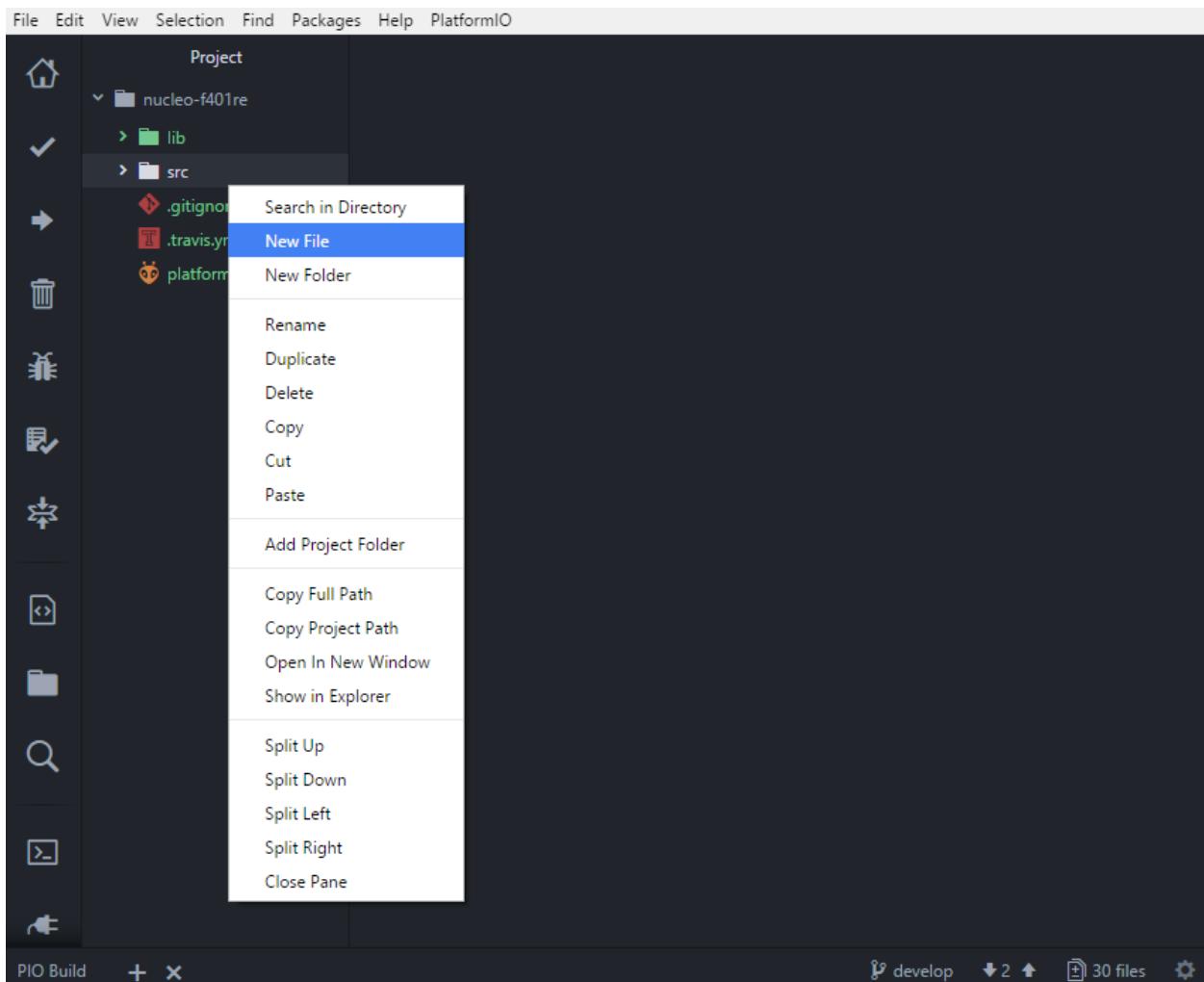
The default framework used with ST Nucleo-F401RE board is *mbed*, but since we decided to use *STM32Cube* we need to change the framework parameter in *Project Configuration File platformio.ini* to the next one:

```
[env:nucleo_f401re]
platform = ststm32
board = nucleo_f401re
framework = stm32cube
```

After these steps, we have a fully configured project that is ready for developing code with *STM32Cube* framework.

Adding Code to the Generated Project

Let's add some actual code to the project. Firstly, we create two main files `main.c` and `main.h` in the `src_dir` folder. Right click on the `src` in the project window:



Add next content to main.h:

```
#ifndef MAIN_H
#define MAIN_H

#include "stm32f4xx_hal.h"

#define LED_PIN          GPIO_PIN_5
#define LED_GPIO_PORT    GPIOA
#define LED_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()

#endif // MAIN_H
```

Add this code to main.c:

```
#include "main.h"

void LED_Init();

int main(void) {
    HAL_Init();
    LED_Init();
```

(continues on next page)

(continued from previous page)

```
while (1)
{
    HAL_GPIO_TogglePin(LED_GPIO_PORT, LED_PIN);
    HAL_Delay(1000);
}

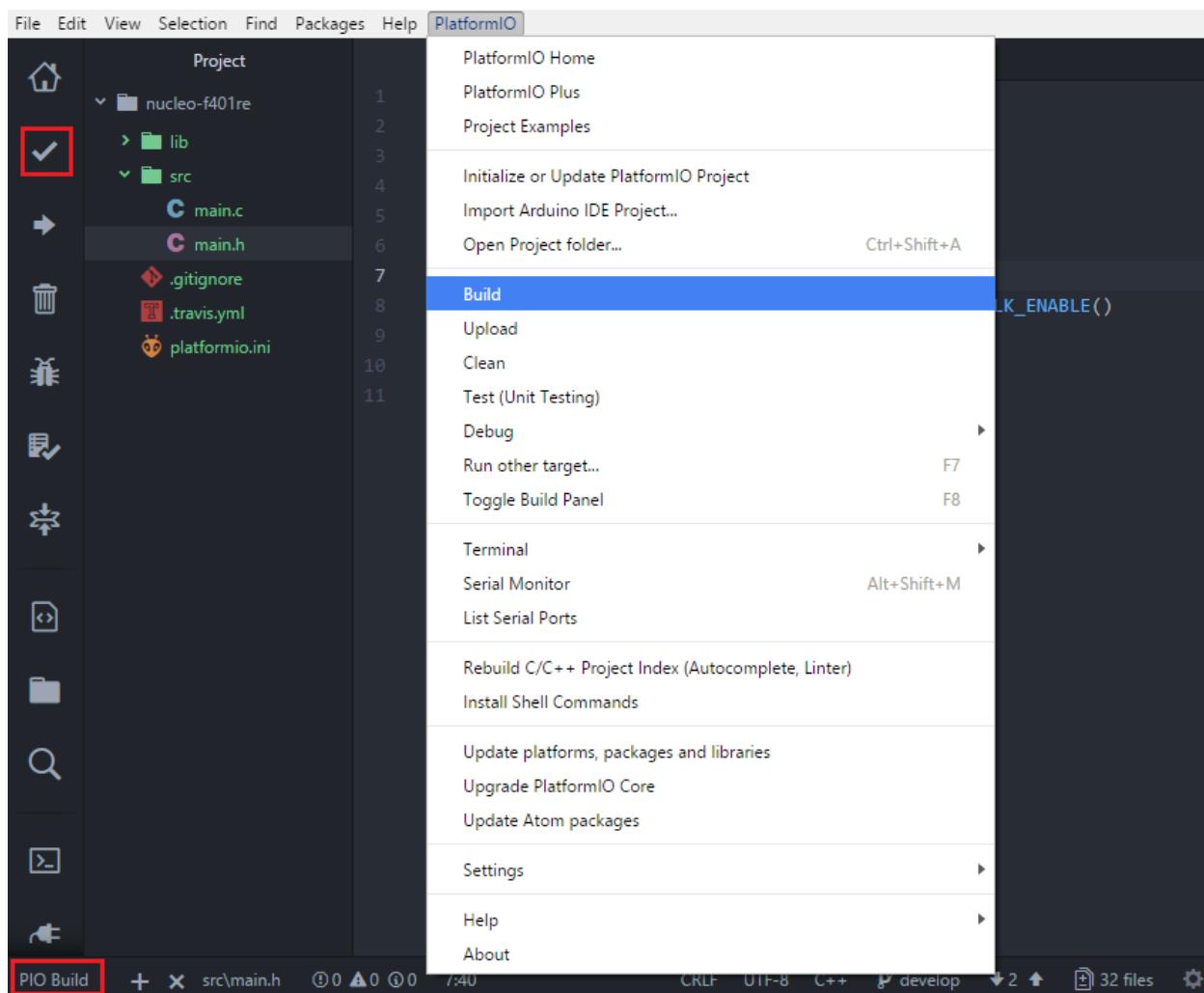
void LED_Init() {
    LED_GPIO_CLK_ENABLE();
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Pin = LED_PIN;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
    HAL_GPIO_Init(LED_GPIO_PORT, &GPIO_InitStruct);
}

void SysTick_Handler(void) {
    HAL_IncTick();
}
```

After this step, we created a basic blink project that is ready for compiling and uploading.

Compiling and Uploading the Firmware

Now we can build the project. To compile firmware we can use three options: Using Build button on *PlatformIO Toolbar*, using Menu: PlatformIO > Build option from top menu, using targets list in bottom left corner or via hotkeys cmd-alt-b / ctrl-alt-b / f9:



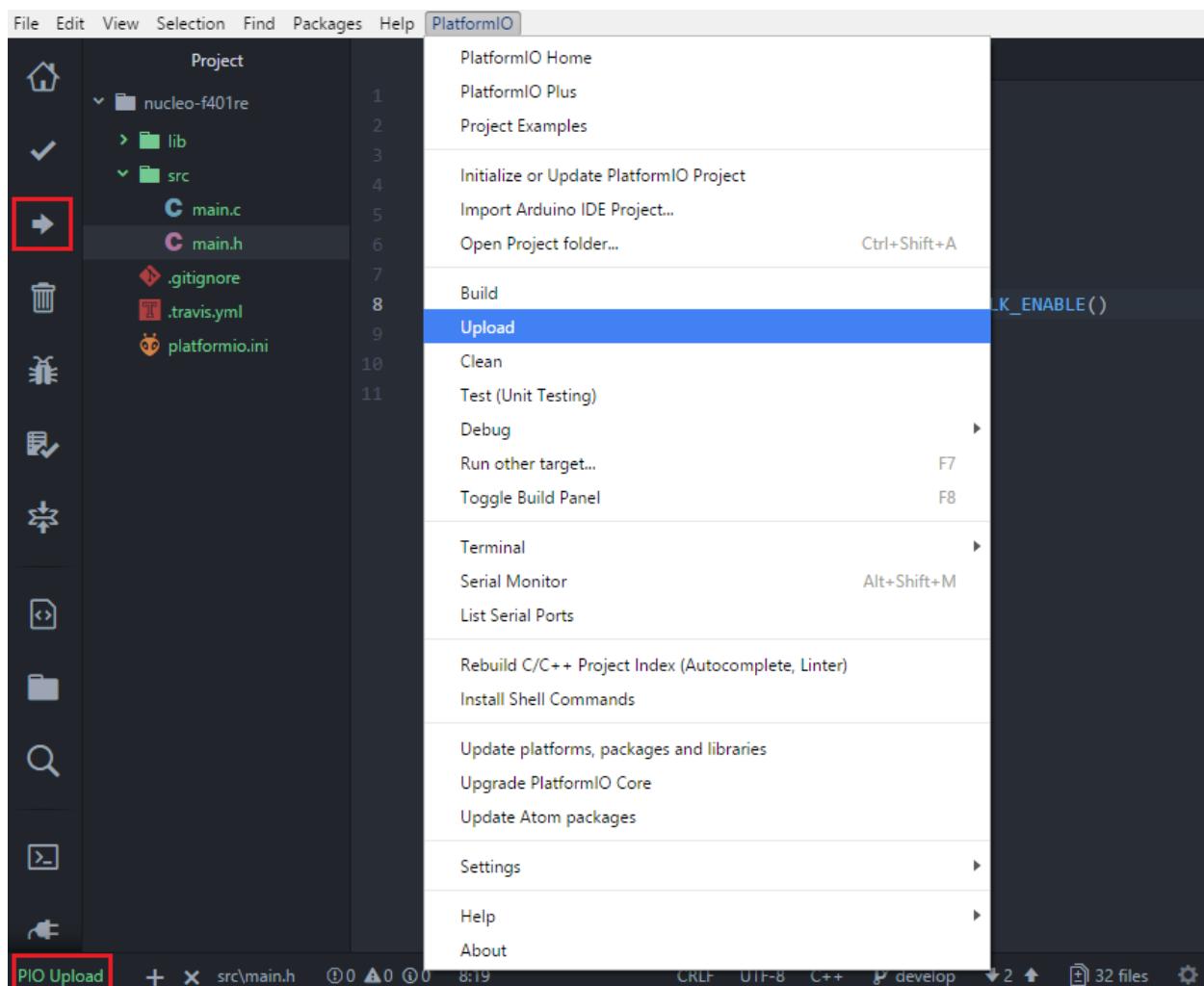
If everything went well, we should see successful result in the terminal window:

The screenshot shows the PlatformIO IDE interface. On the left is a sidebar with various icons for file operations like creating, deleting, and saving files. The main area has a dark background. At the top, there's a menu bar with File, Edit, View, Selection, Find, Packages, Help, and PlatformIO. Below the menu is a toolbar with icons for project navigation, file operations, and build status. The central workspace shows a project structure for 'nucleo-f401re'. Under 'src', there are 'main.c' and 'main.h' files. The code editor shows the content of 'main.h'. The bottom right corner of the editor has a green bar with the text 'platformio run' and a timer showing '2.1 s'. Below the editor is a terminal window with the following output:

```
platformio run
Looking for dependencies...
Project does not have dependencies
Calculating size .pioenvs\nucleo_f401re\firmware.elf
text      data      bss      dec      hex filename
1768      1080     1568     4416    1140 .pioenvs\nucleo_f401re\firmware.elf
[SUCCESS] Took 1.82 seconds
```

At the bottom of the interface, there's a footer with buttons for PIO Build, a file list, and other settings.

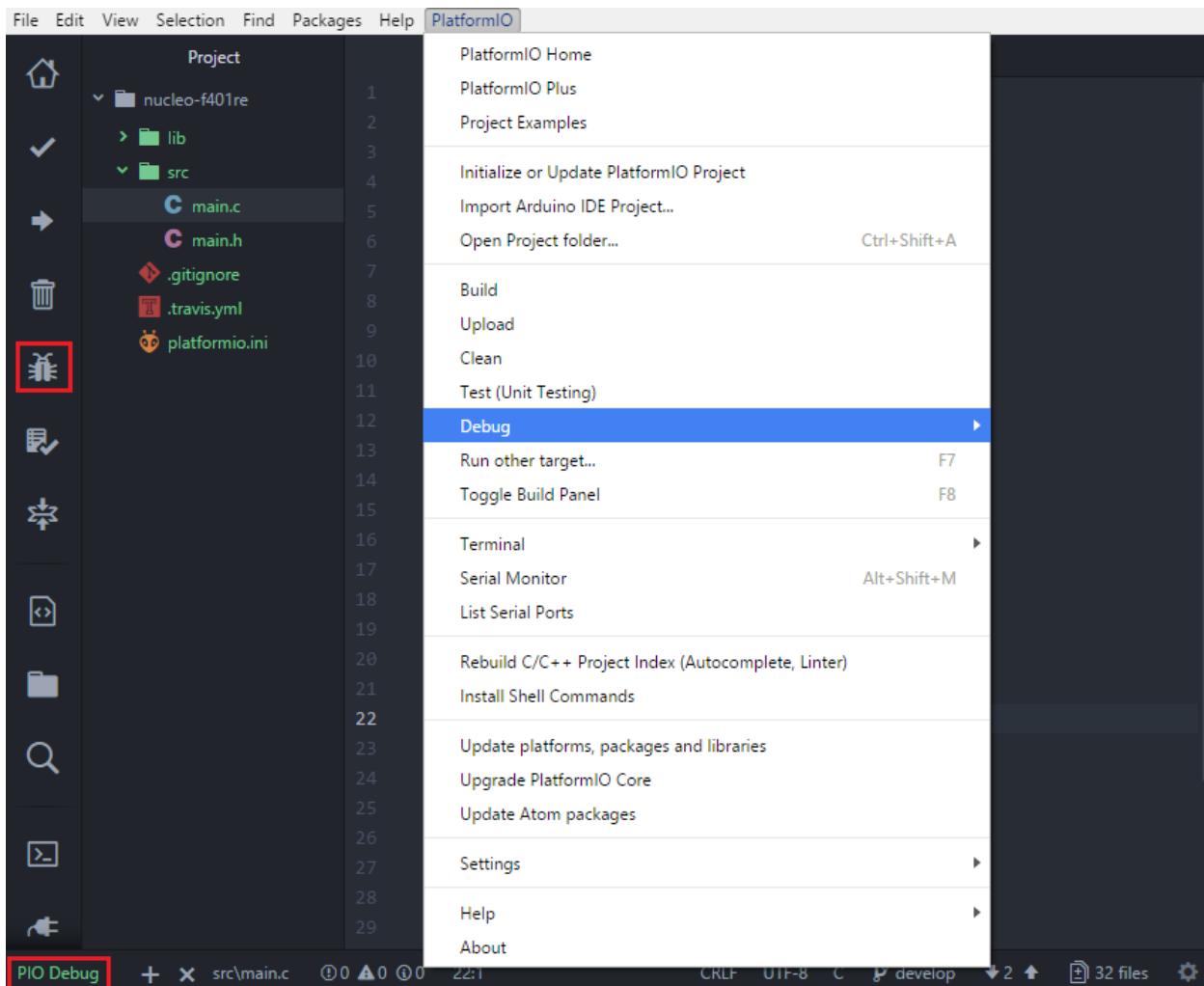
Now we can upload firmware to the board: Using Build button on *PlatformIO Toolbar*, using Menu: PlatformIO > Upload from top menu, using targets list in bottom left corner or via hotkeys cmd+alt+u / ctrl+alt+u



After successful uploading, the green LED2 should start blinking.

Debugging the Firmware

PIO Unified Debugger offers the easiest way to debug your board. Just click Debug button on *PlatformIO Toolbar* or use Menu: PlatformIO > Debug > Start new debug session:



We need to wait some time while PlatformIO is initializing debug session and when the first line after the main function is highlighted we are ready to debug:

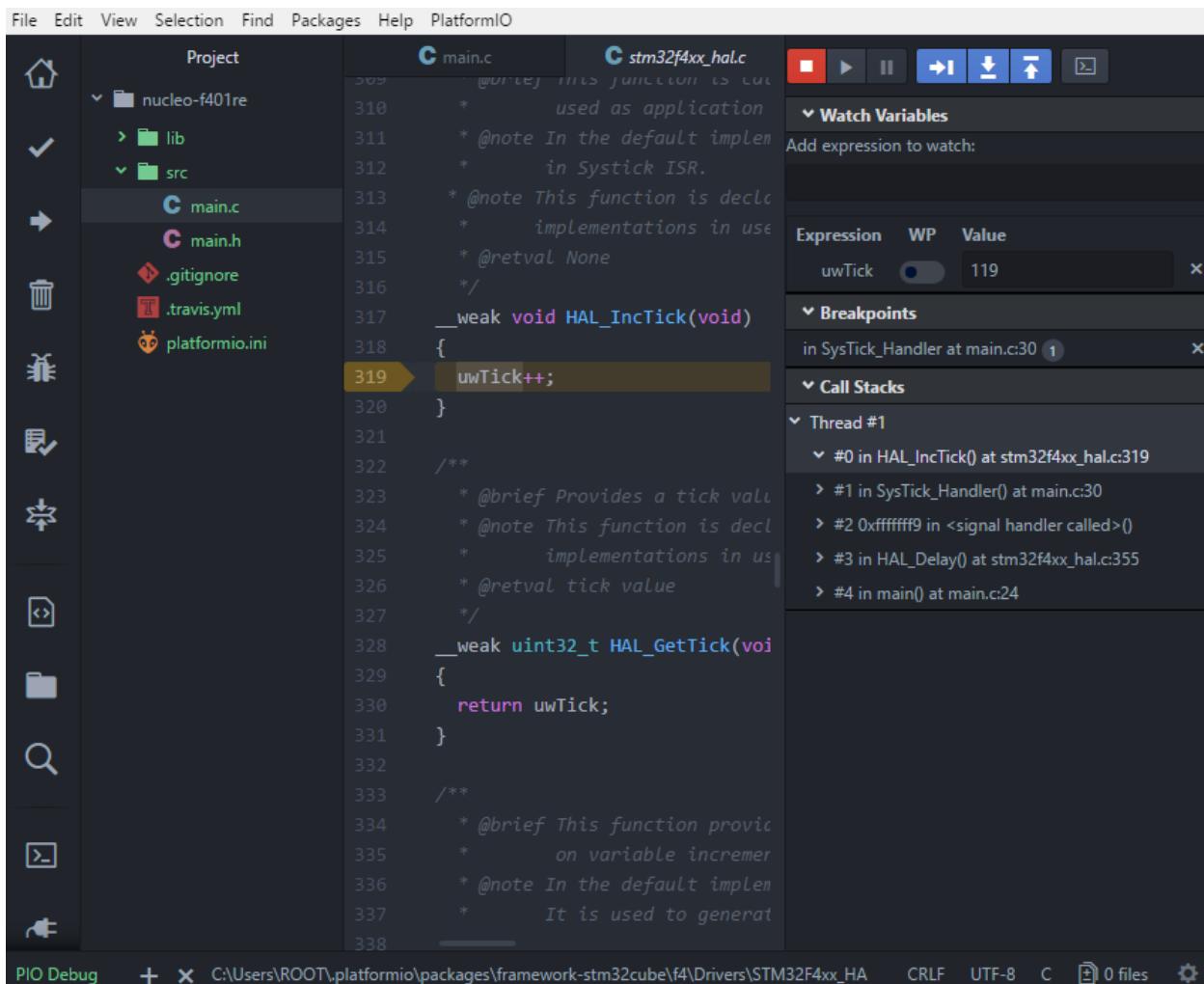
The screenshot shows the PlatformIO IDE interface. On the left is a sidebar with various icons for file operations. The central area is a code editor displaying the `src/main.c` file. The code initializes GPIO for an LED and toggles it in a loop. A yellow arrow points to line 18, where a breakpoint is set. The right side of the interface contains several windows: a toolbar with run/debug buttons, a "Watch Variables" window, a "Breakpoints" window, a "Call Stacks" window, and a "Thread #1" window showing the current execution point.

```

4   {
5     LED_GPIO_CLK_ENABLE();
6
7     GPIO_InitTypeDef GPIO_InitStruct;
8
9     GPIO_InitStruct.Pin = LED_PIN;
10    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
11    GPIO_InitStruct.Pull = GPIO_PULLDOWN;
12    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
13    HAL_GPIO_Init(LED_GPIO_PORT, &GPIO_InitStruct);
14 }
15
16 int main(void)
17 {
18 //> HAL_Init();
19 //> LED_Init();
20
21     while (1)
22     {
23         HAL_GPIO_TogglePin(LED_GPIO_PORT);
24         HAL_Delay(1000);
25     }
26 }
27
28 void SysTick_Handler(void)
29 {
30     HAL_IncTick();
31 }

```

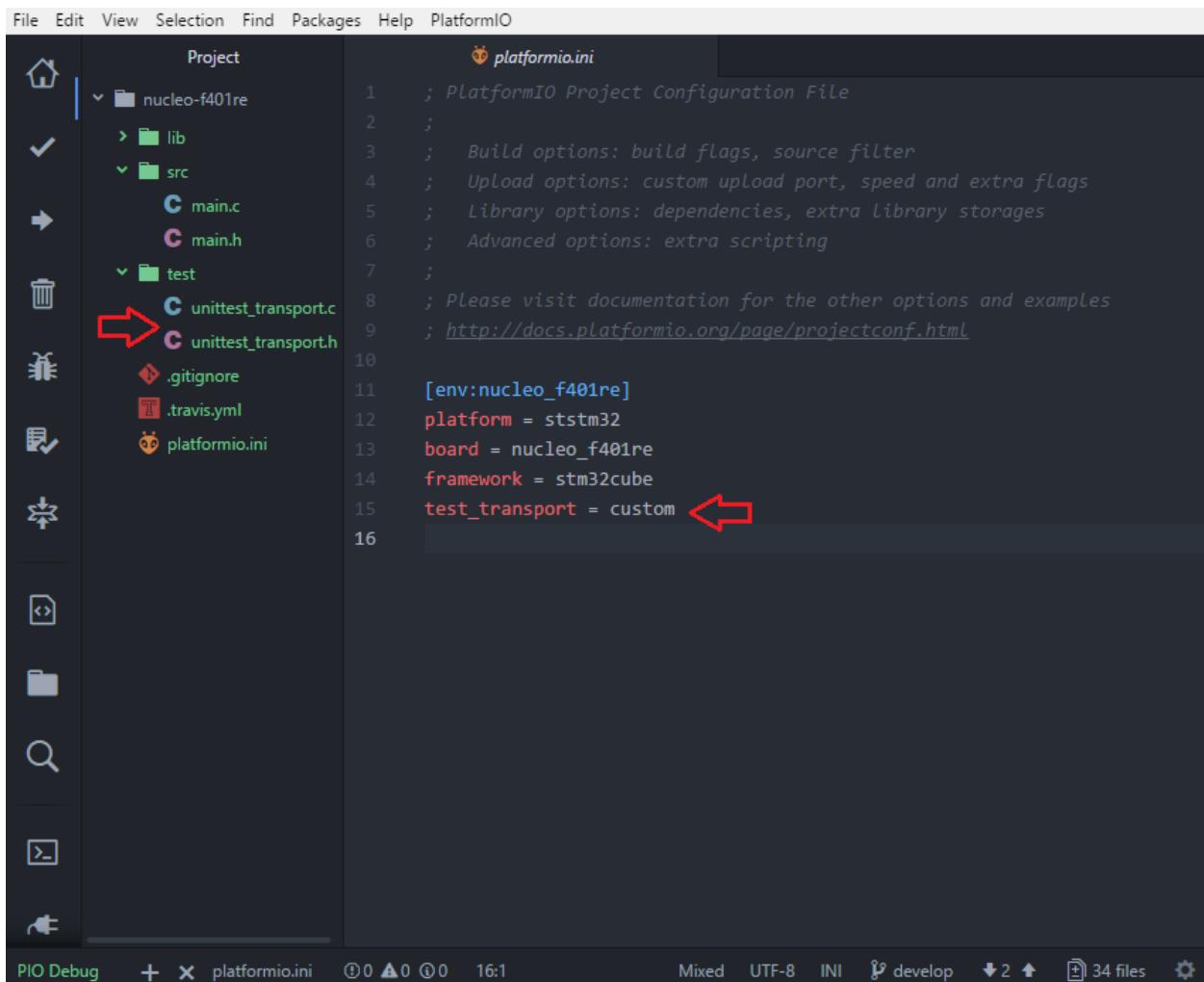
We can walk through the code using control buttons, set breakpoints, add variables to Watch window:



Writing Unit Tests

Now let's write some tests using [PIO Unit Testing](#) feature that can help us test code directly on the target board. [PIO Unit Testing](#) engine by default supports only three frameworks: [Arduino](#), [Energia](#) and [mbed](#). Since we decided to use [STM32Cube](#) we need to implement a custom [test_transport](#) to print testing results and specify that condition in [Project Configuration File](#) `platformio.ini`:

```
[env:nucleo_f401re]
platform = ststm32
board = nucleo_f401re
framework = stm32cube
test_transport = custom
```



We will use USART2 on ST Nucleo-F401RE board because it's directly connected to the STLink debug interface and in OS it can be visible as a Virtual Com Port, so we don't need any additional USB-UART converter. To implement the custom `test_transport` we need to create two files `unittest_transport.h` and `unittest_transport.c` and put them in the `test_dir` in the root folder of our project. In these files we need to implement next four functions:

```

void unittest_uart_begin();
void unittest_uart_putchar(char c);
void unittest_uart_flush();
void unittest_uart_end();

```

Implementation of `unittest_transport.h`:

```

#ifndef UNITTEST_TRANSPORT_H
#define UNITTEST_TRANSPORT_H

#ifdef __cplusplus
extern "C" {
#endif

void unittest_uart_begin();
void unittest_uart_putchar(char c);
void unittest_uart_flush();

```

(continues on next page)

(continued from previous page)

```
void unittest_uart_end();

#ifndef __cplusplus
}
#endif

#endif // UNITEST_TRANSPORT_H
```

Implementation of unittest_transport.c:

```
#include "unittest_transport.h"
#include "stm32f4xx_hal.h"

#define USARTx
#define USARTx_CLK_ENABLE()
#define USARTx_CLK_DISABLE()
#define USARTx_RX_GPIO_CLK_ENABLE()
#define USARTx_TX_GPIO_CLK_ENABLE()
#define USARTx_RX_GPIO_CLK_DISABLE()
#define USARTx_TX_GPIO_CLK_DISABLE()

#define USARTx_FORCE_RESET()
#define USARTx_RELEASE_RESET()

#define USARTx_TX_PIN
#define USARTx_TX_GPIO_PORT
#define USARTx_TX_AF
#define USARTx_RX_PIN
#define USARTx_RX_GPIO_PORT
#define USARTx_RX_AF

static UART_HandleTypeDef UartHandle;

void unittest_uart_begin()
{
    GPIO_InitTypeDef GPIO_InitStruct;

    USARTx_TX_GPIO_CLK_ENABLE();
    USARTx_RX_GPIO_CLK_ENABLE();

    USARTx_CLK_ENABLE();

    GPIO_InitStruct.Pin      = USARTx_TX_PIN;
    GPIO_InitStruct.Mode     = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull     = GPIO_PULLUP;
    GPIO_InitStruct.Speed    = GPIO_SPEED_FAST;
    GPIO_InitStruct.Alternate = USARTx_TX_AF;

    HAL_GPIO_Init(USARTx_TX_GPIO_PORT, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = USARTx_RX_PIN;
    GPIO_InitStruct.Alternate = USARTx_RX_AF;

    HAL_GPIO_Init(USARTx_RX_GPIO_PORT, &GPIO_InitStruct);
    UartHandle.Instance     = USARTx;
    UartHandle.Init.BaudRate = 115200;
```

(continues on next page)

(continued from previous page)

```

UartHandle.Init.WordLength      = UART_WORDLENGTH_8B;
UartHandle.Init.StopBits       = UART_STOPBITS_1;
UartHandle.Init.Parity         = UART_PARITY_NONE;
UartHandle.Init.HwFlowCtl      = UART_HWCONTROL_NONE;
UartHandle.Init.Mode           = UART_MODE_TX_RX;
UartHandle.Init.OverSampling   = UART_OVERSAMPLING_16;

if(HAL_UART_Init(&UartHandle) != HAL_OK) {
    while(1) {}
}

void unittest_uart_putchar(char c)
{
    HAL_UART_Transmit(&UartHandle, (uint8_t*)(&c), 1, 1000);
}

void unittest_uart_flush() {}

void unittest_uart_end() {
    USARTx_CLK_DISABLE();
    USARTx_RX_GPIO_CLK_DISABLE();
    USARTx_TX_GPIO_CLK_DISABLE();
}

```

Now we need to add some test cases. Tests can be added to a single C file that may include multiple tests. First of all, in this file we need to add three default functions: `setUp`, `tearDown` and `main`. `setUp` and `tearDown` are used to initialize and finalize test conditions. Implementations of these functions are not required for running tests but if you need to initialize some variables before you run a test, you use the `setUp` function and if you need to clean up variables you use `tearDown` function. In our example we will use these functions to accordingly initialize and deinitialize LED. `main` function acts as a simple program where we describe our test plan.

Let's implement some basic tests for blinking routine:

```

#include <main.h>
#include <unity.h>

void setUp(void) {
    HAL_Init();

    LED_GPIO_CLK_ENABLE();

    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Pin = LED_PIN;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
    HAL_GPIO_Init(LED_GPIO_PORT, &GPIO_InitStruct);
}

void tearDown(void) {
    HAL_GPIO_DeInit(LED_GPIO_PORT, LED_PIN);
}

void test_led_builtin_pin_number(void) {

```

(continues on next page)

(continued from previous page)

```
TEST_ASSERT_EQUAL(LED_PIN, GPIO_PIN_5);

}

void test_led_state_high(void) {
    HAL_GPIO_WritePin(LED_GPIO_PORT, LED_PIN, GPIO_PIN_SET);
    TEST_ASSERT_EQUAL(HAL_GPIO_ReadPin(LED_GPIO_PORT), LED_PIN), GPIO_PIN_SET);
}

void test_led_state_low(void) {
    HAL_GPIO_WritePin(LED_GPIO_PORT, LED_PIN, GPIO_PIN_RESET);
    TEST_ASSERT_EQUAL(HAL_GPIO_ReadPin(LED_GPIO_PORT), LED_PIN), GPIO_PIN_RESET);
}

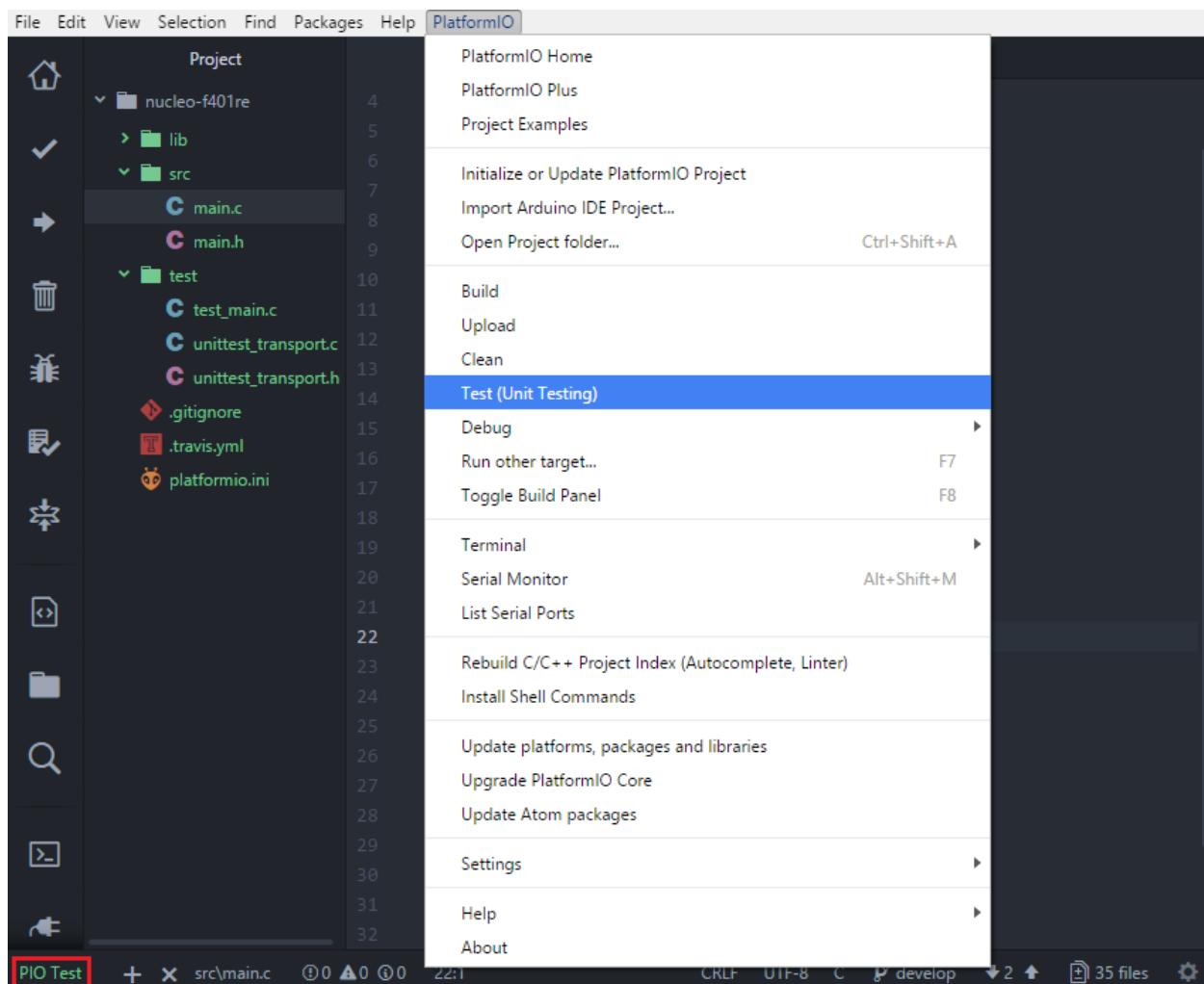
int main() {
    UNITY_BEGIN();
    RUN_TEST(test_led_builtin_pin_number);

    for (unsigned int i = 0; i < 5; i++)
    {
        RUN_TEST(test_led_state_high);
        HAL_Delay(500);
        RUN_TEST(test_led_state_low);
        HAL_Delay(500);
    }

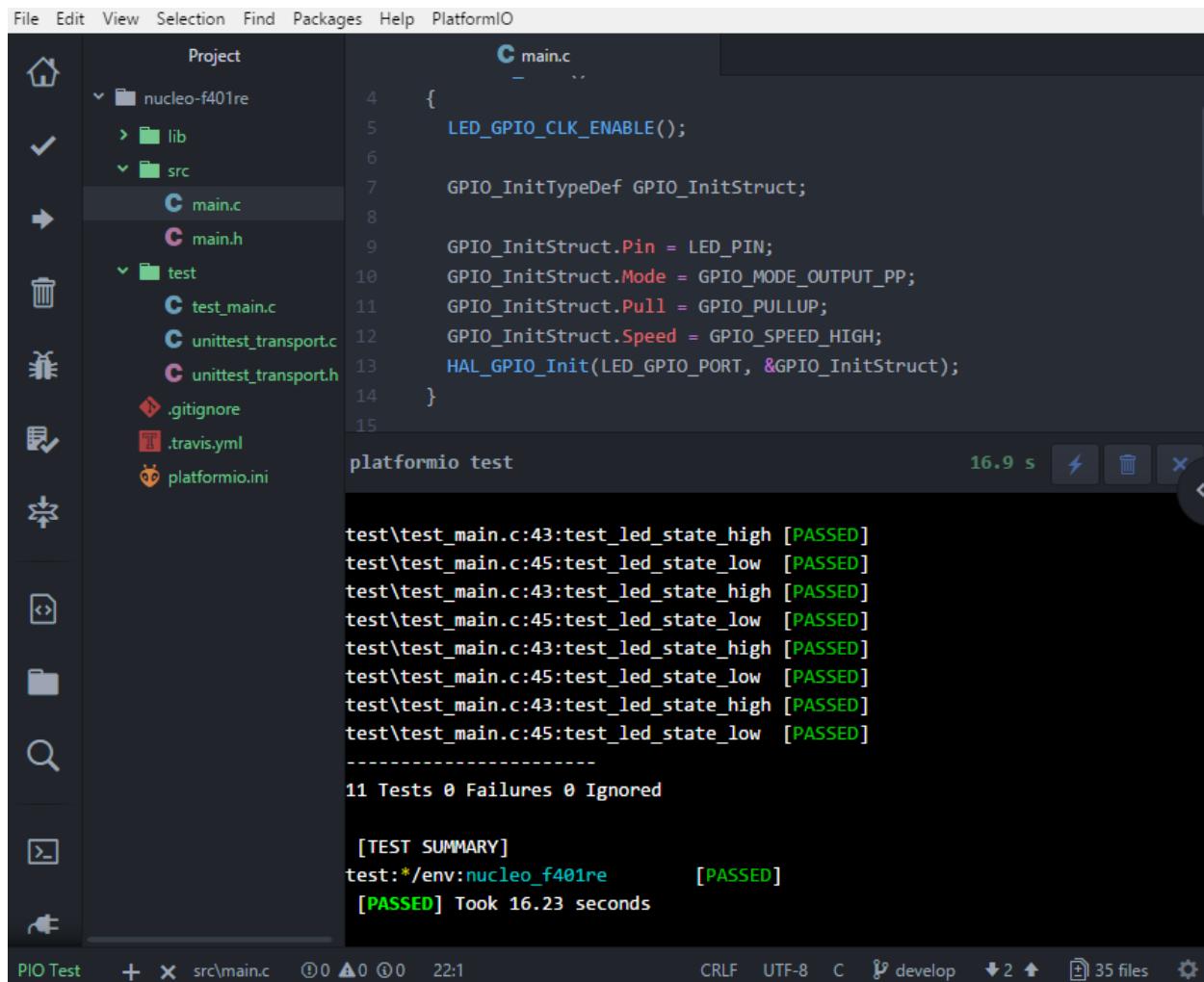
    UNITY_END(); // stop unit testing

    while(1) {}
}
```

Now we are ready to upload tests to the board. To do this we can use Menu: PlatformIO > Test (Unit Testing) from top menu or targets list in bottom left corner:



After processing we should see a detailed report about testing results:



The screenshot shows the PlatformIO IDE interface within VSCode. On the left is the Project Explorer with files like main.c, main.h, test_main.c, and test_unittest_transport.c. The main area shows the code for main.c and a terminal window displaying test results. The terminal output is as follows:

```

C main.c
4 {
5     LED_GPIO_CLK_ENABLE();
6
7     GPIO_InitTypeDef GPIO_InitStruct;
8
9     GPIO_InitStruct.Pin = LED_PIN;
10    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
11    GPIO_InitStruct.Pull = GPIO_PULLUP;
12    GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
13    HAL_GPIO_Init(LED_GPIO_PORT, &GPIO_InitStruct);
14 }
15

platformio test
16.9 s ⚡ 🗑 ✖️ ↻

test\test_main.c:43:test_led_state_high [PASSED]
test\test_main.c:45:test_led_state_low [PASSED]
test\test_main.c:43:test_led_state_high [PASSED]
test\test_main.c:45:test_led_state_low [PASSED]
test\test_main.c:43:test_led_state_high [PASSED]
test\test_main.c:45:test_led_state_low [PASSED]
test\test_main.c:43:test_led_state_high [PASSED]
test\test_main.c:45:test_led_state_low [PASSED]
-----
11 Tests 0 Failures 0 Ignored

[TEST SUMMARY]
test:/env:nucleo_f401re [PASSED]
[PASSED] Took 16.23 seconds

```

At the bottom, the status bar shows "PIO Test" and "src\main.c".

Congratulations! As we can see from the report, all our tests went successfully!

Conclusion

Now we have a decent template that we can improve for our next more complex projects.

Project Source Code

The source code of this tutorial is available at <https://github.com/platformio/platformio-examples/tree/develop/unit-testing/stm32cube>

Arduino and Nordic nRF52-DK: debugging and unit testing

The goal of this tutorial is to demonstrate how simple it is to use *PlatformIO IDE for VSCode* to develop, run and debug a simple project with *Arduino* framework for Nordic nRF52-DK board.

- **Level:** Beginner
- **Platforms:** Windows, Mac OS X, Linux

Requirements:

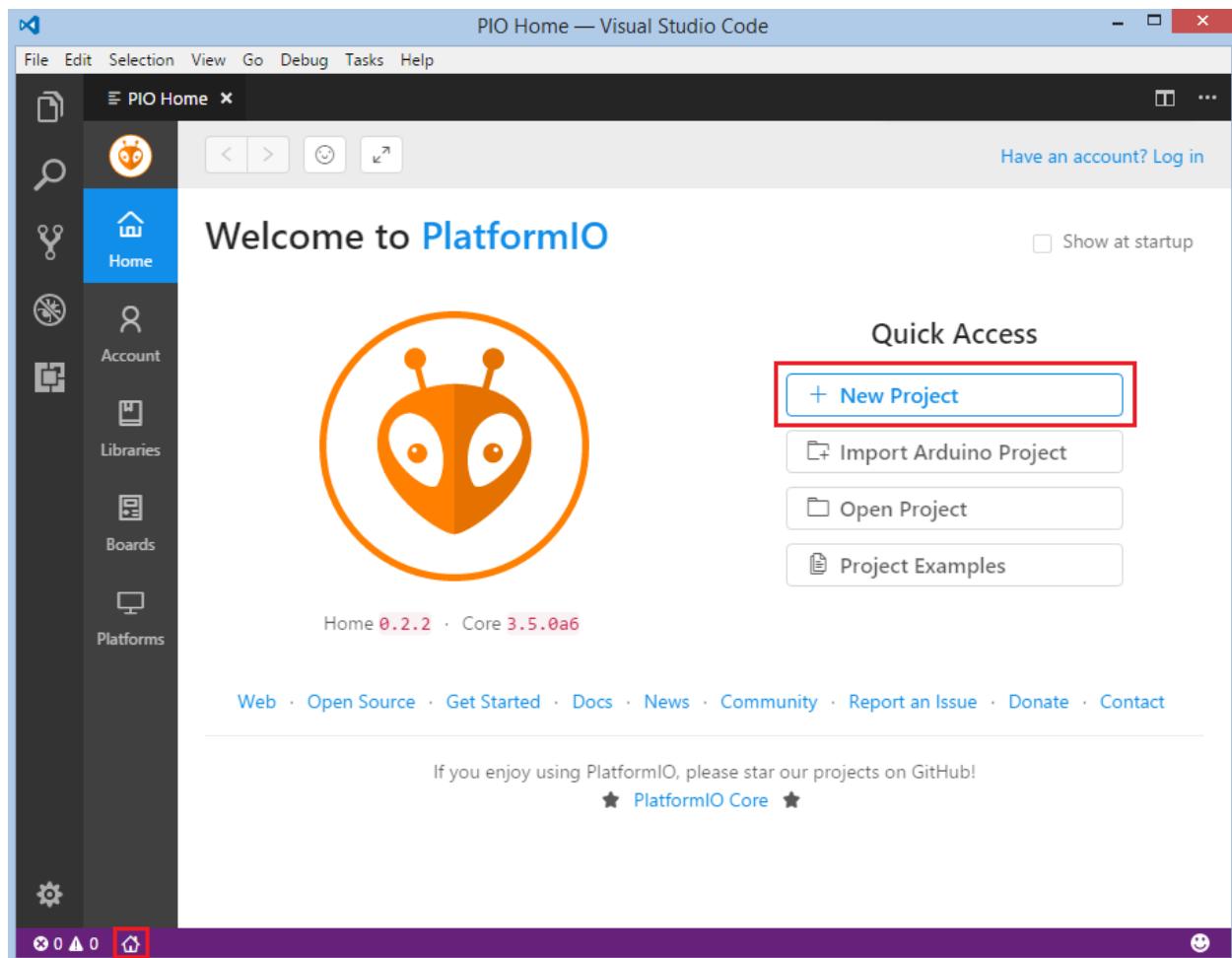
- Downloaded and installed *PlatformIO IDE for VSCode*
- Nordic nRF52-DK development board

Contents

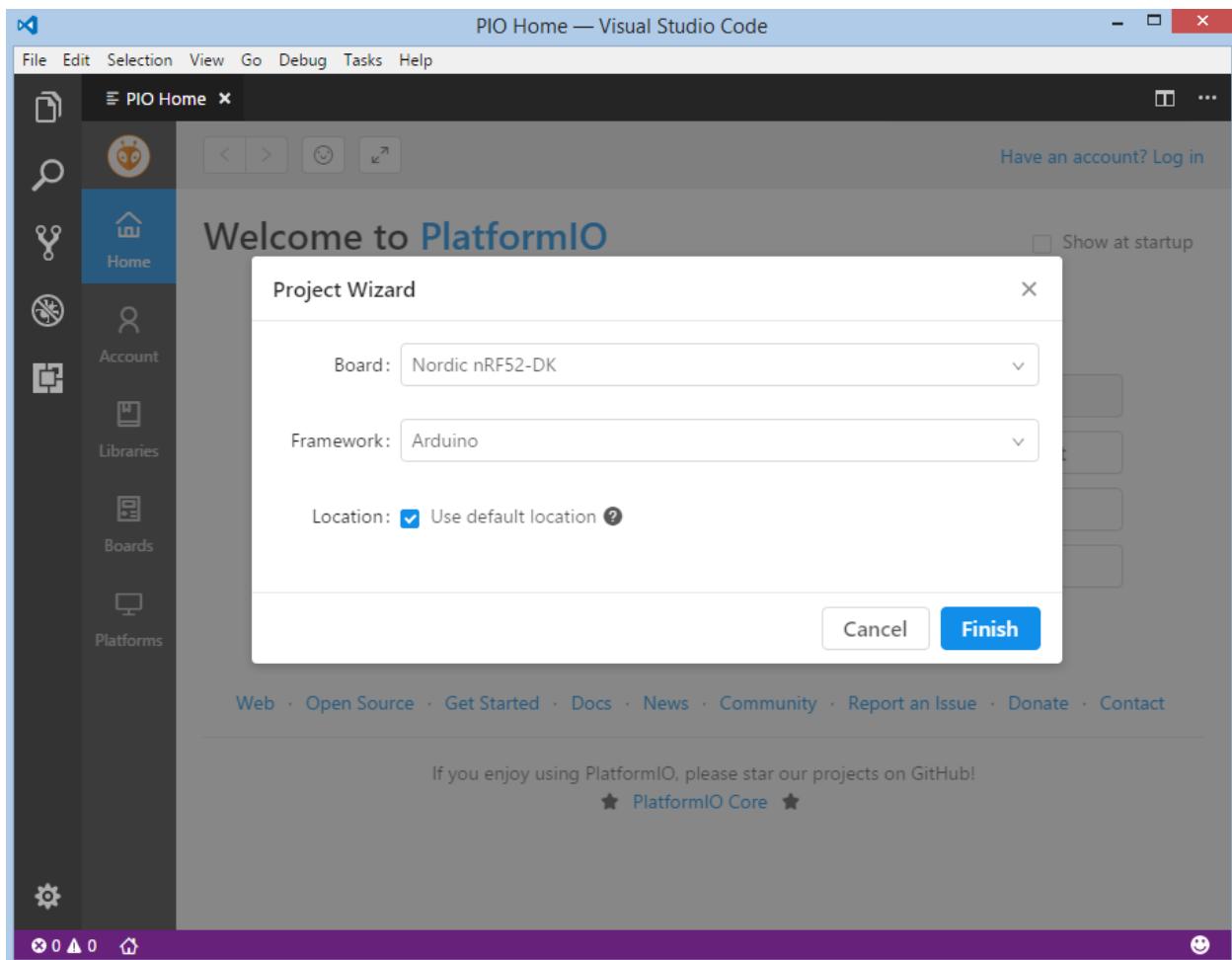
- *Setting Up the Project*
- *Adding Code to the Generated Project*
- *Compiling and Uploading the Firmware*
- *Debugging the Firmware*
- *Writing Unit Tests*
- *Adding Bluetooth LE features*
- *Conclusion*

Setting Up the Project

At first step, we need to create a new project using PlatformIO Home Page (to open this page just press Home icon on the toolbar):



On the next step we need to select Nordic nRF52-DK as a development board, *Arduino* as a framework and a path to the project location (or use the default one):



Processing the selected project may take some amount of time (PlatformIO will download and install all required packages) and after these steps, we have a fully configured project that is ready for developing code with *Arduino* framework.

Adding Code to the Generated Project

Let's add some actual code to the project. Firstly, we open a default main file named `main.cpp` in the `src_dir` folder and replace its content with next one:

```
#include <Arduino.h>

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

void loop()
{
    digitalWrite(LED_BUILTIN, HIGH);
    delay(100);
    digitalWrite(LED_BUILTIN, LOW);
    delay(100);
}
```

The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar (Explorer) displays a project structure for a '170901-181544-NRF52_DK' board. The 'src' folder contains a file named 'main.cpp', which is currently selected and highlighted with a red border. The main editor window shows the following C++ code:

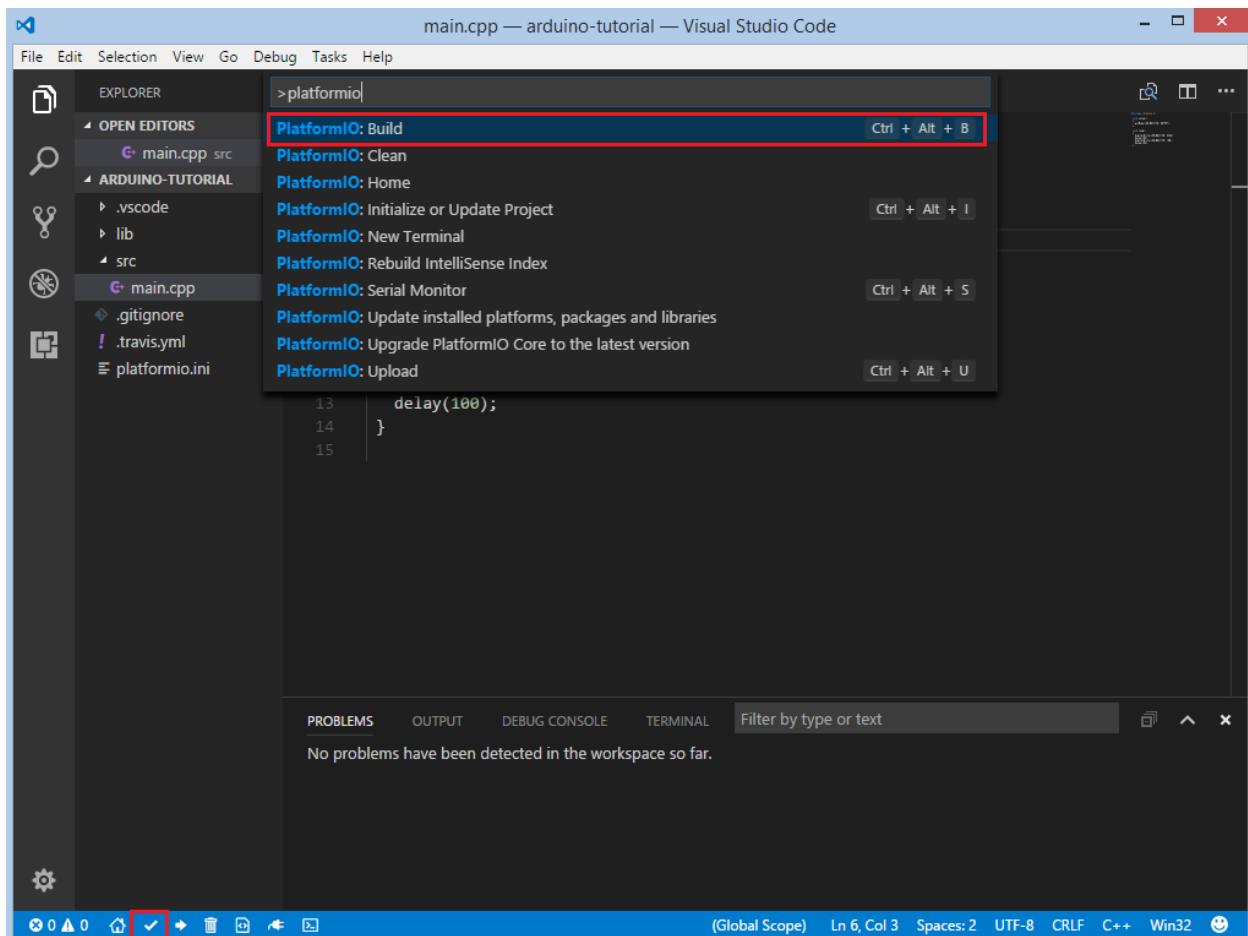
```
#include <Arduino.h>
void setup()
{
    pinMode(LED_BUILTIN, OUTPUT);
}
void loop()
{
    digitalWrite(LED_BUILTIN, HIGH);
    delay(100);
    digitalWrite(LED_BUILTIN, LOW);
    delay(100);
}
```

The status bar at the bottom provides build information: (Global Scope) Ln 15, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32.

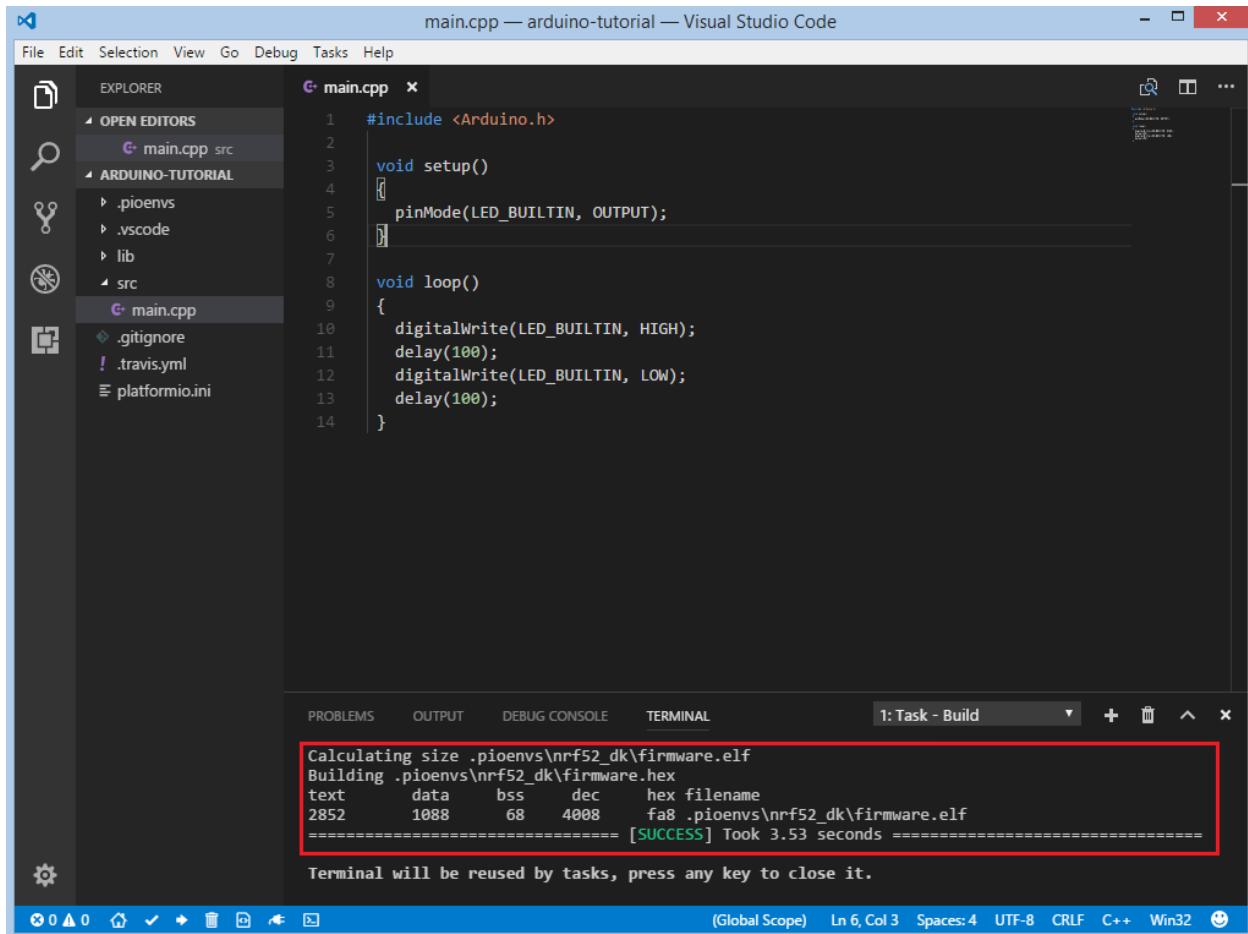
After this step, we created a basic blink project ready for compiling and uploading.

Compiling and Uploading the Firmware

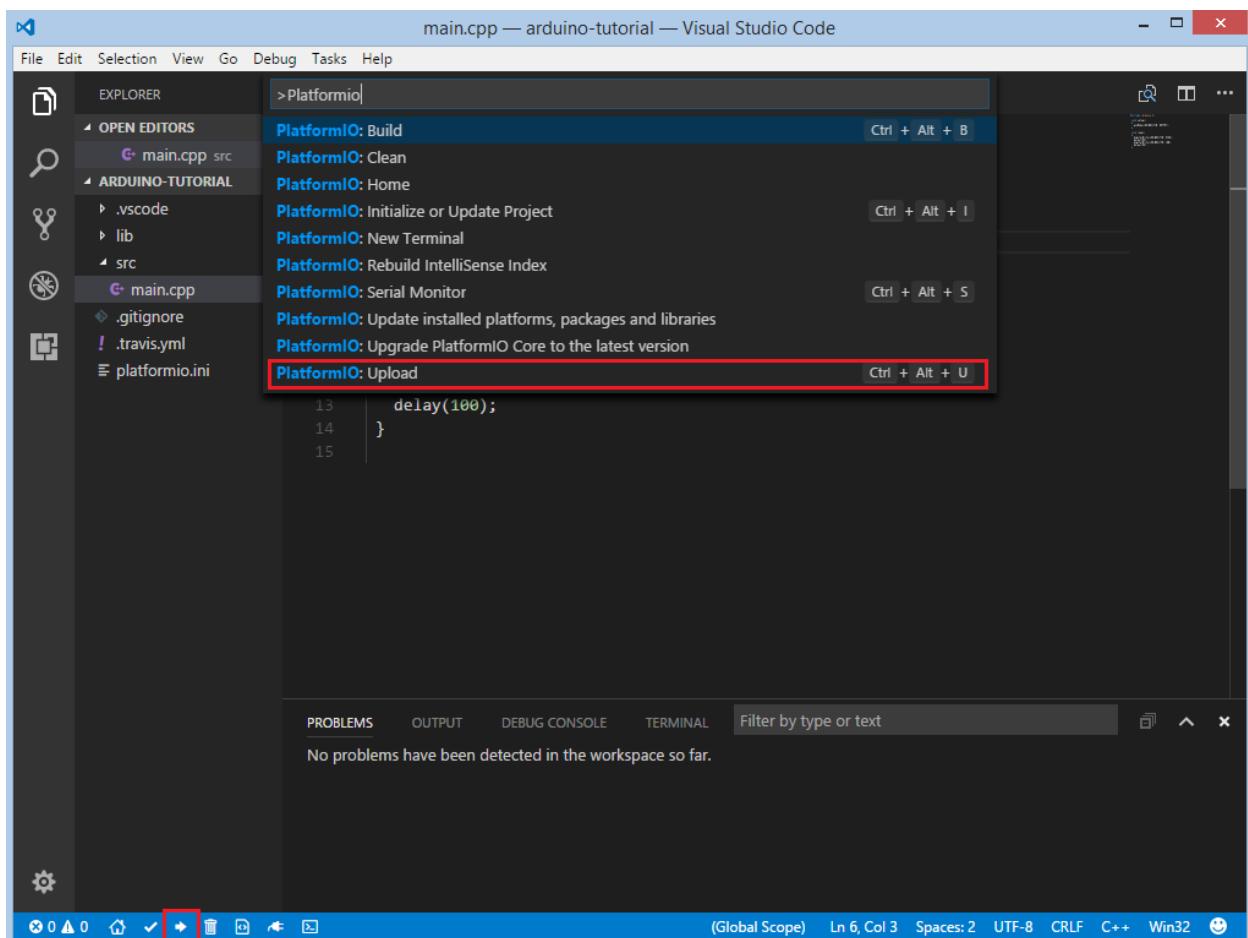
Now we can build the project. To compile firmware we can use three options: Using Build button on *PlatformIO Toolbar*, using Command Palette View: Command Palette > PlatformIO: Build, using Task Menu Tasks: Run Task... > PlatformIO: Build or via hotkeys cmd-alt-b / ctrl-alt-b:



If everything went well, we should see a successful result message in the terminal window:



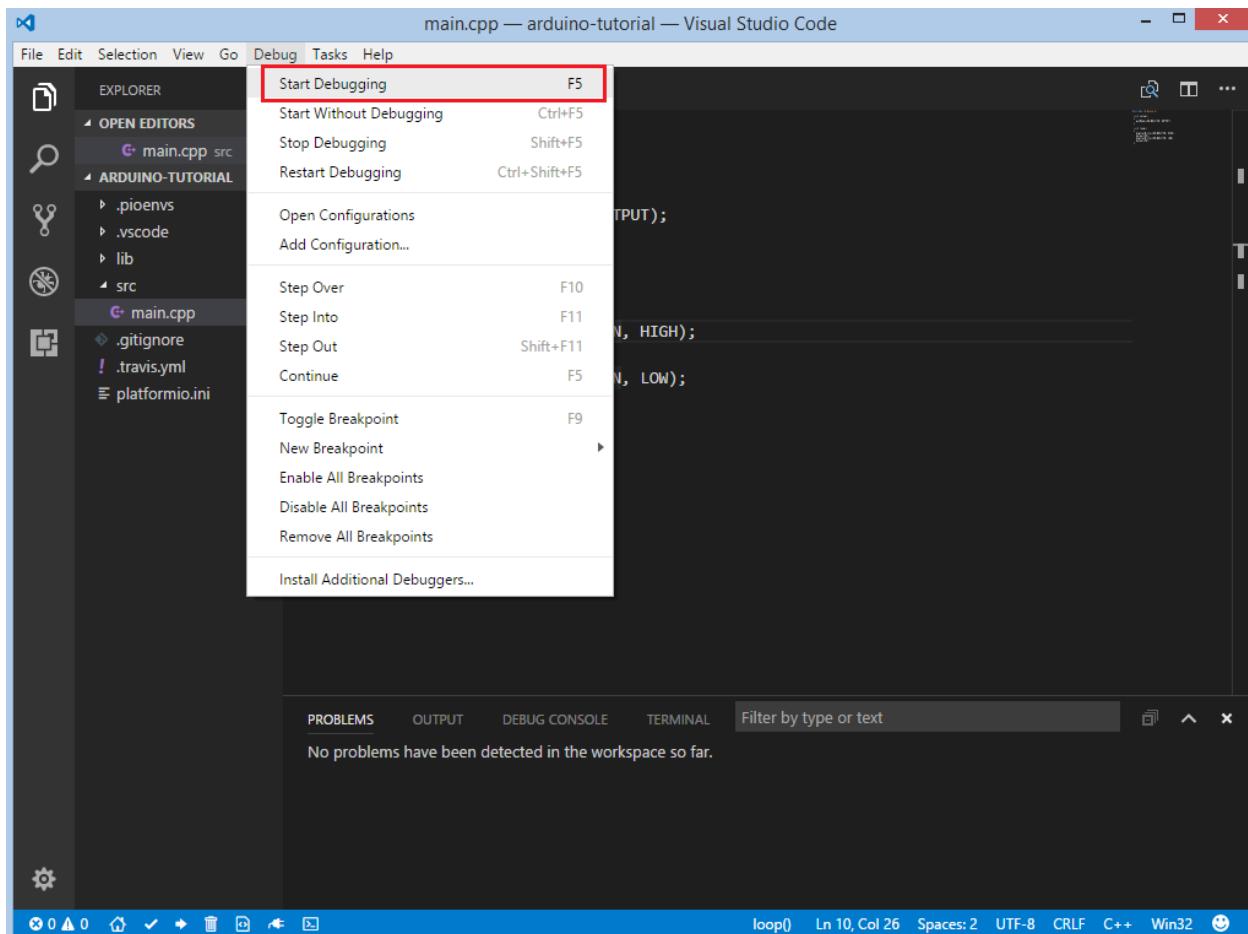
Now we can upload firmware to the board: Using Upload button on *PlatformIO Toolbar*, using Command Palette View: Command Palette > PlatformIO: Upload, using Task Menu Tasks: Run Task... > PlatformIO: Upload or via hotkeys cmd+alt+u / ctrl+alt+u:



After successful uploading, the green LED1 should start blinking.

Debugging the Firmware

PIO Unified Debugger offers the easiest way to debug your board. Just navigate to the top menu and select Debug: Start debugging or use hotkey button F5:



We need to wait some time while PlatformIO is initializing debug session and when the first line after the main function is highlighted we are ready to debug:

The screenshot shows the Visual Studio Code interface with the Arduino extension. The main editor window displays the `main.cpp` file for the `arduino-tutorial` project. The code is as follows:

```
#define ARDUINO_MAIN
#include "Arduino.h"

// Weak empty variant initialization function.
// May be redefined by variant files.
void initVariant() __attribute__((weak));
void initVariant() { }

/*
 * \brief Main entry point of Arduino application
 */
int main( void )
{
    init();

    initVariant();

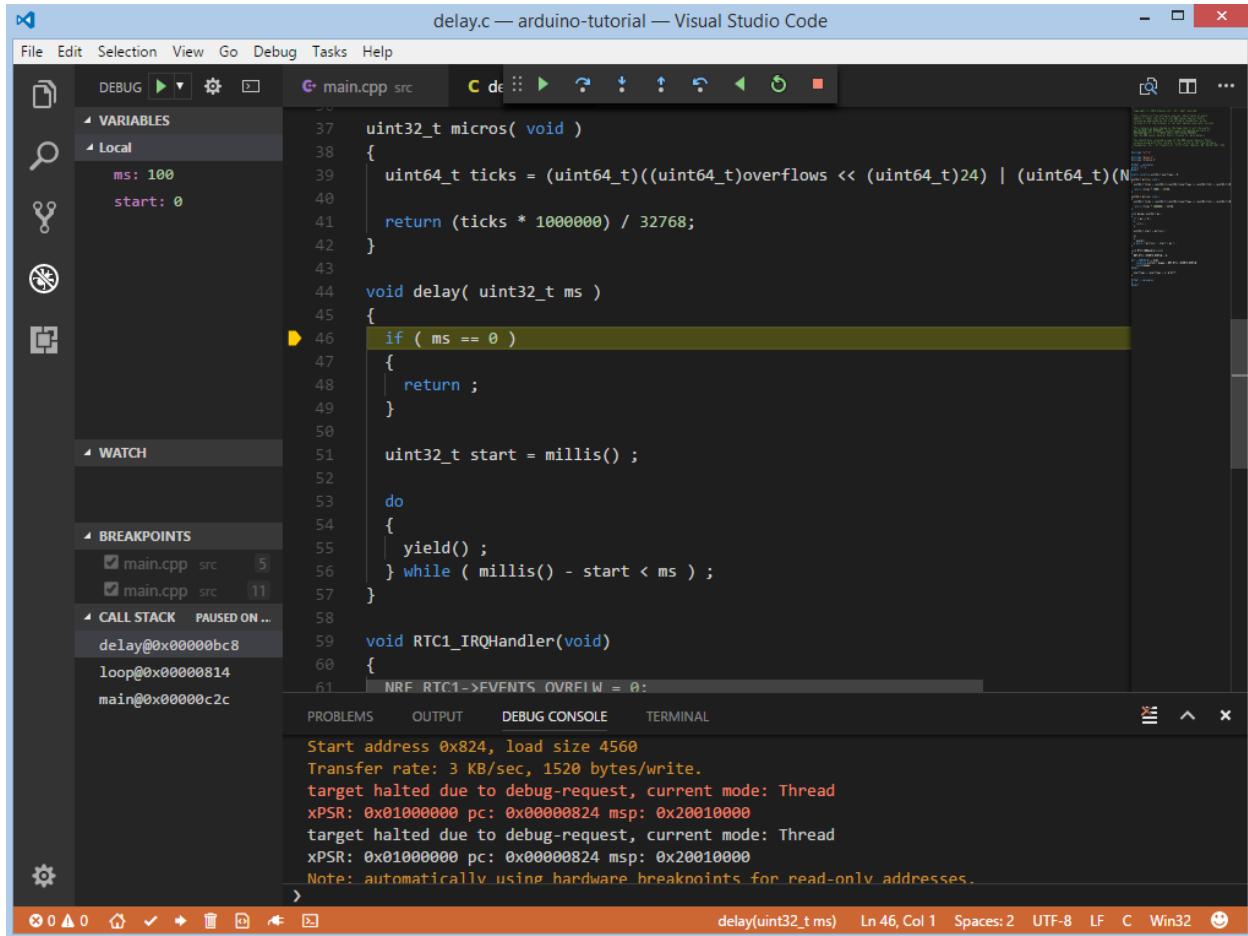
    delay(1);

    setup();

    for (;;)
    {
        loop();
        if (serialEventRun) serialEventRun();
    }
}
```

The code editor has a green bar highlighting the opening brace of the `main` function. The left sidebar contains the **VARIABLES**, **Local**, **WATCH**, **CALL STACK PAUSED ON...** (showing `main@0x0000c14`), and **BREAKPOINTS** sections. The bottom status bar shows the current file is `main(void)`, line 28, column 1, with settings for Spaces: 2, LF, C++, Win32.

We can walk through the code using control buttons, set breakpoints, add variables to Watch window:



Writing Unit Tests

Test cases can be added to a single file that may include multiple tests. First of all, in this file, we need to add three default functions: `setUp`, `tearDown`, `setup` and `loop`. `setUp` and `tearDown` are used to initialize and finalize test conditions. Implementations of these functions are not required for running tests but if you need to initialize some variables before you run a test, you use the `setUp` function and if you need to clean up variables you use `tearDown` function. In our example we will use these functions to accordingly initialize and deinitialize LED. `setup` and `loop` functions act as a simple Arduino program where we describe our test plan.

Let's implement some basic tests for blinking routine:

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

// void setUp(void) {
// // set stuff up here
// }

// void tearDown(void) {
// // clean stuff up here
// }

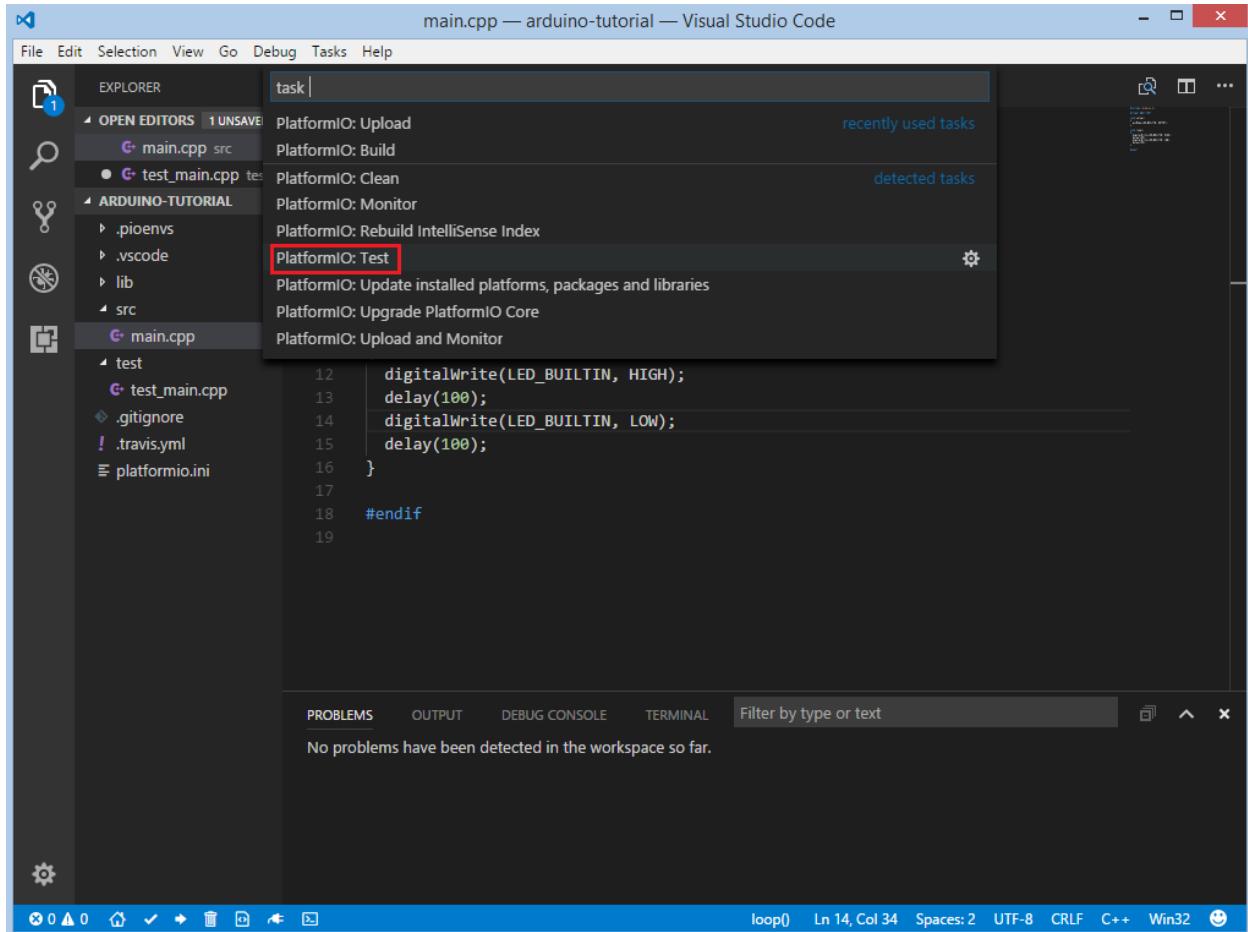
void test_led_builtin_pin_number(void)
```

(continues on next page)

(continued from previous page)

```
{  
    TEST_ASSERT_EQUAL(LED_BUILTIN, 13);  
  
}  
  
void test_led_state_high(void)  
{  
    digitalWrite(LED_BUILTIN, HIGH);  
    TEST_ASSERT_EQUAL(digitalRead(LED_BUILTIN), LOW);  
}  
  
void test_led_state_low(void)  
{  
    digitalWrite(LED_BUILTIN, LOW);  
    TEST_ASSERT_EQUAL(digitalRead(LED_BUILTIN), LOW);  
}  
  
void setup()  
{  
    UNITY_BEGIN();  
    RUN_TEST(test_led_builtin_pin_number);  
    pinMode(LED_BUILTIN, OUTPUT);  
  
    for (uint8_t i = 0; i < 5; i++)  
    {  
        RUN_TEST(test_led_state_high);  
        delay(200);  
        RUN_TEST(test_led_state_low);  
        delay(200);  
    }  
  
    UNITY_END(); // stop unit testing  
}  
  
void loop()  
{  
}
```

Now we are ready to upload tests to the board. To do this we can use Tasks: Run Task... > PlatformIO Test from top menu:



After processing we should see a detailed report about testing results:

```

test_main.cpp — arduino-tutorial — Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER main.cpp test_main.cpp
OPEN EDITORS main.cpp src test_main.cpp test
ARDUINO-TUTORIAL .pioenvs .vscode lib src main.cpp test test_main.cpp .gitignore .travis.yml platformio.ini
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 3: Task - Test + - ×
=====
[test::*] Testing... (3/3)
If you don't see any output for the first 10 secs, please reset board (press reset button)

SS
test\test_main.cpp:39:test_led_state_high      [PASSED]
test\test_main.cpp:41:test_led_state_low       [PASSED]
test\test_main.cpp:39:test_led_state_high      [PASSED]
test\test_main.cpp:41:test_led_state_low       [PASSED]
test\test_main.cpp:39:test_led_state_high      [PASSED]
test\test_main.cpp:41:test_led_state_low       [PASSED]
-----
11 Tests 1 Failures 0 Ignored

[TEST SUMMARY]
test:/env:nrf52_dk      [PASSED]
===== [PASSED] Took 5.65 seconds =====
Terminal will be reused by tasks, press any key to close it.
setup() Ln 35, Col 34 Spaces: 4 UTF-8 C++ Win32

```

As we can see from the report, all our tests were successful!

Adding Bluetooth LE features

To add the basic BLE functionality to our project we need to define the SoftDevice version and install a library called `BLEPeripheral`. Both these modifications can be specified in *Project Configuration File* `platformio.ini`:

```
[env:nrf52_dk]
platform = nordicnrf52
board = nrf52_dk
framework = arduino
; SoftDevice version
build_flags = -DNRF52_S132
lib_deps =
    BLEPeripheral
```

Now let's create a basic application that can interact with other BLE devices (e.g. phone). For example, next code declares a BLE characteristic that controls the state of the LED1

```
#include <Arduino.h>
#include <SPI.h>
#include <BLEPeripheral.h>

BLEPeripheral ledPeripheral = BLEPeripheral();
```

(continues on next page)

(continued from previous page)

```
BLEService ledService = BLEService("19b10000e8f2537e4f6cd104768a1214");
BLECharCharacteristic ledCharacteristic =_
~BLECharCharacteristic("19b10001e8f2537e4f6cd104768a1214", BLERead | BLEWrite);

void setup()
{
    pinMode(LED_BUILTIN, OUTPUT);

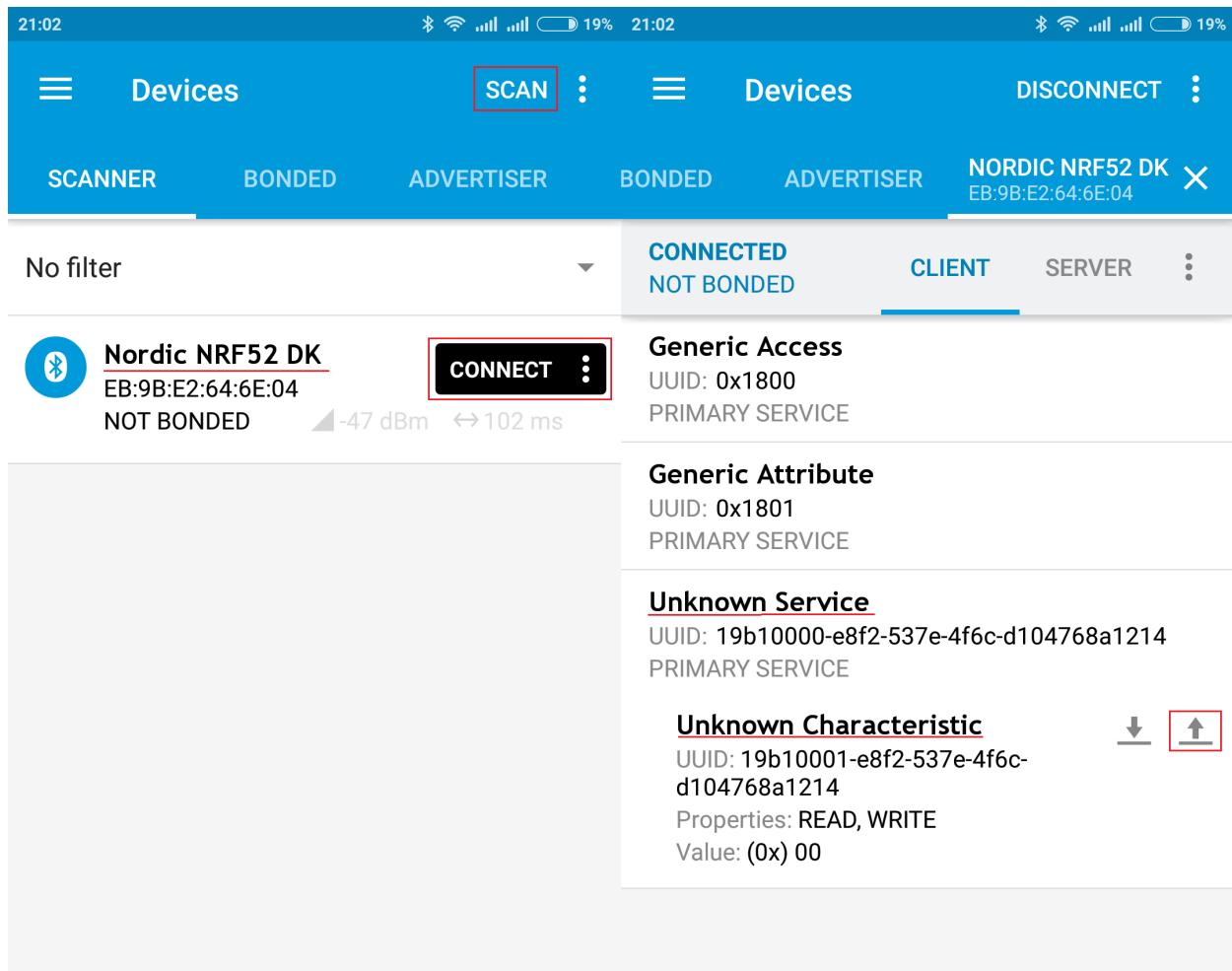
    ledPeripheral.setAdvertisedServiceUuid(ledService.uuid());
    ledPeripheral.addAttribute(ledService);
    ledPeripheral.addAttribute(ledCharacteristic);
    ledPeripheral.setLocalName("Nordic NRF52 DK");
    ledPeripheral.begin();
}

void loop()
{
    BLECentral central = ledPeripheral.central();

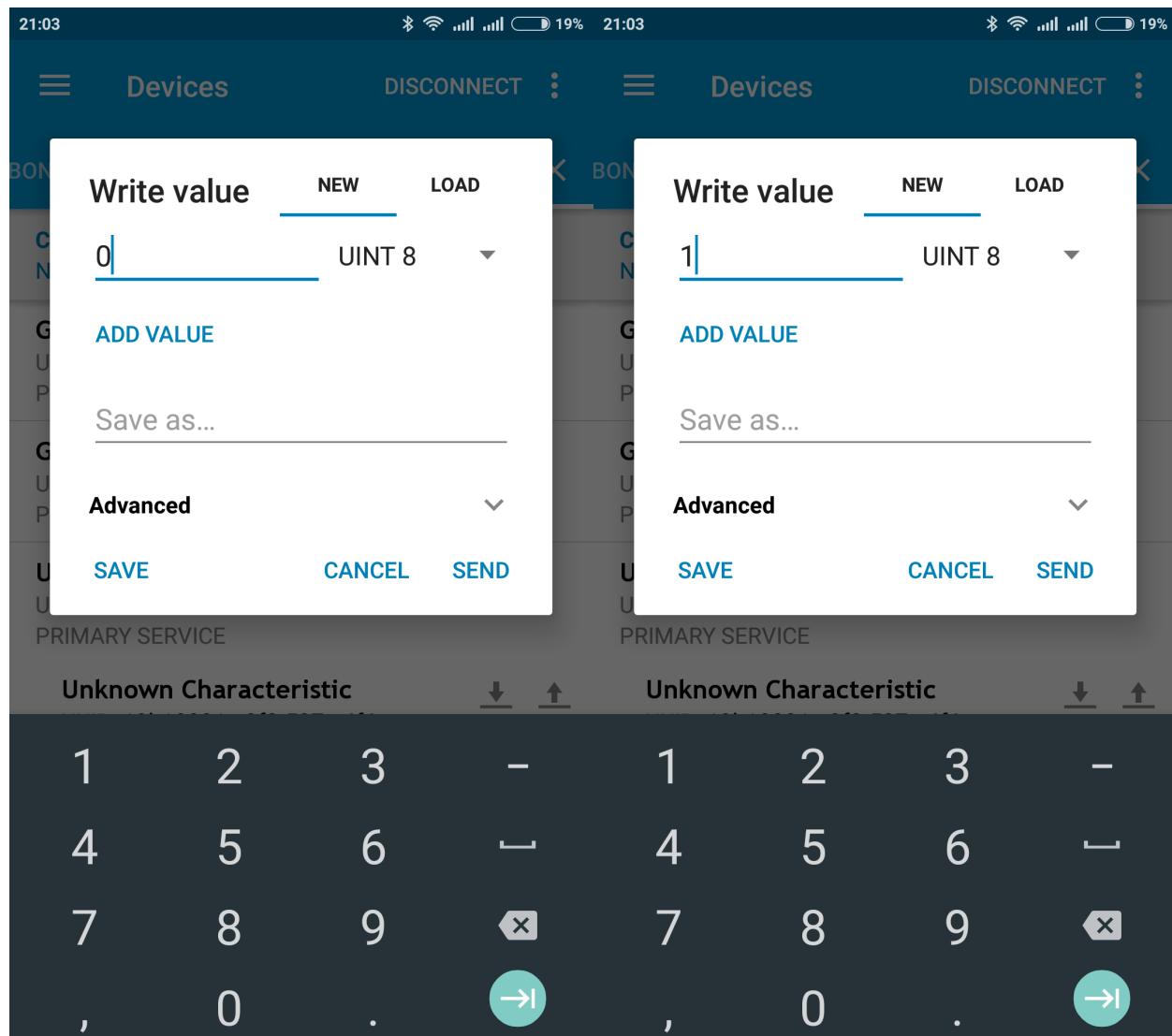
    if (central) {
        while (central.connected()) {
            if (ledCharacteristic.written()) {
                if (ledCharacteristic.value()) {
                    digitalWrite(LED_BUILTIN, HIGH);
                }
                else{
                    digitalWrite(LED_BUILTIN, LOW);
                }
            }
        }
    }
}
```

Now we can compile and upload this program to the board as described in previous sections. To verify that our application works as expected, we can use any Android smartphone with BLE feature and [Nordic nRF Connect tool](#).

At first, we need to scan all advertising BLE devices and connect to the device called `Nordic NRF52 DK`. After a successful connection to the board, we should see one “Unknown Service” with one “Unknown Characteristic” fields:



To switch the LED on or off we just need write 0 or 1 as `UINT8` to the BLE characteristic:



Conclusion

Now we have a project template for Nordic nRF52-DK board that we can use as a boilerplate for the next projects.

1.5.2 Project Examples

Pre-configured projects with source code are located in [PlatformIO Examples repository](#).

1.6 Project Configuration File `platformio.ini`

The Project configuration file is named `platformio.ini`. This is a [INI-style](#) file.

`platformio.ini` has sections (each denoted by a `[header]`) and key / value pairs within the sections. Lines beginning with `;` are ignored and may be used to provide comments.

Multi-values option could be specified in 2 ways:

1. Split values with “,” (comma + space)
2. Use multi-line format, where each new line should start with 2 spaces (minimum)

There are 2 system reserved sections:

- *PlatformIO Core* settings: `Section [platformio]`
- Environment settings: `Section [env:NAME]`

The other sections can be used by users, for example, for *Dynamic variables*. The sections and their allowable values are described below.

1.6.1 Section `[platformio]`

- *Options*
 - `description`
 - `env_default`
 - `home_dir`
 - `build_dir`
 - `include_dir`
 - `src_dir`
 - `lib_dir`
 - `libdeps_dir`
 - `lib_extra_dirs`
 - `data_dir`
 - `test_dir`
 - `boards_dir`

A `platformio` section is used for overriding default configuration options for *PlatformIO Core*.

Note: Relative path is allowed for directory option:

- ~ will be expanded to user's home directory
 - .. / or .. \ go up to one folder
-

Options

`description`

Describe a project with a short information. PlatformIO uses it for *PlatformIO Home* in the multiple places.

env_default

`platformio run` command processes all environments [env:***] by default if `platformio run --environment` option is not specified. `env_default` allows to define environments which should be processed by default.

Multiple environments are allowed if they *are separated with “, ” (comma+space)*. For example.

```
[platformio]
env_default = uno, nodemcu

[env:uno]
platform = atmelavr
framework = arduino
board = uno

[env:nodemcu]
platform = espressif8266
framework = arduino
board = nodemcu

[env:teensy31]
platform = teensy
framework = arduino
board = teensy31

[env:lpmsp430g2553]
platform = tmsp430
framework = energia
board = lpmsp430g2553
build_flags = -D LED_BUILTIN=RED_LED
```

home_dir

Is used to store platform toolchains, frameworks, global libraries for *Library Dependency Finder (LDF)*, service data and etc. The size of this folder will depend on number of installed development platforms.

A default value is User's home directory:

- Unix `~/.platformio`
- Windows `%HOMEPATH%\platformio`

This option can be overridden by global environment variable `PLATFORMIO_HOME_DIR`.

Example:

```
[platformio]
home_dir = /path/to/custom/pio/storage
```

build_dir

Warning: PLEASE DO NOT EDIT FILES IN THIS FOLDER. PlatformIO will overwrite your changes on the next build. **THIS IS A CACHE DIRECTORY.**

PlatformIO Build System uses this folder for project environments to store compiled object files, static libraries, firmwares and other cached information. It allows PlatformIO to build source code extremely fast!

You can delete this folder without any risk! If you modify *Project Configuration File* `platformio.ini`, then PlatformIO will remove this folder automatically. It will be created on the next build operation.

A default value is `.pioenvs` that means that folder is located in the root of project.

This option can be overridden by global environment variable `PLATFORMIO_BUILD_DIR`.

Note: If you have any problems with building your Project environments which are defined in *Project Configuration File* `platformio.ini`, then **TRY TO DELETE** this folder. In this situation you will remove all cached files without any risk.

`include_dir`

A path to project's headers files. PlatformIO uses it for `platformio run` command. A default value is `include` that means that folder is located in the root of project. This path will be added to `CPPPATH` of build environment.

This option can be overridden by global environment variable `PLATFORMIO_INCLUDE_DIR`.

`src_dir`

A path to project's source directory. PlatformIO uses it for `platformio run` command. A default value is `src` that means that folder is located in the root of project.

This option can be overridden by global environment variable `PLATFORMIO_SRC_DIR`.

Note: This option is useful for people who migrate from Arduino/Energia IDEs where source directory should have the same name like the main source file. See `example` project with own source directory.

`lib_dir`

You can put here your own/private libraries. The source code of each library should be placed in separate directory, like `lib/private_lib/` [here are source files]. This directory has the highest priority for *Library Dependency Finder (LDF)*.

A default value is `lib` that means that folder is located in the root of project.

This option can be overridden by global environment variable `PLATFORMIO_LIB_DIR`.

For example, see how can be organized `Foo` and `Bar` libraries:

```
|--lib
|   |--Bar
|   |   |--docs
|   |   |--examples
|   |   |--src
|   |       |- Bar.c
|   |       |- Bar.h
|   |--Foo
|       |- Foo.c
```

(continues on next page)

(continued from previous page)

```
| | | - Foo.h  
| - platformio.ini  
| --src  
|   | - main.c
```

Then in `src/main.c` you should use:

```
#include <Foo.h>  
#include <Bar.h>  
  
// rest H/C/CPP code
```

PlatformIO will find your libraries automatically, configure preprocessor's include paths and build them.

libdeps_dir

Internal storage where *Library Manager* will install project dependencies (`lib_deps`). A default value is `.` `piolibdeps` that means that folder is located in the root of project.

This option can be overridden by global environment variable `PLATFORMIO_LIBDEPS_DIR`.

lib_extra_dirs

New in version 3.2.

A list with extra storages for a project where *Library Dependency Finder (LDF)* will look for libraries.

This option has the same behavior as `lib_extra_dirs` option for a specific build environment defined in `[env:]` section. The main difference is that the option which is defined in `[platformio]` section will be extra applied automatically for all `[env:]` sections.

For the possible values and examples please follow to `lib_extra_dirs`.

data_dir

Data directory to store contents and *Uploading files to file system SPIFFS*. A default value is `data` that means that folder is located in the root of project.

This option can be overridden by global environment variable `PLATFORMIO_DATA_DIR`.

test_dir

Directory where *PIO Unit Testing* engine will look for the tests. A default value is `test` that means that folder is located in the root of project.

This option can be overridden by global environment variable `PLATFORMIO_TEST_DIR`.

boards_dir

Custom board settings per project. You can change this path with your own. A default value is `boards` that means that folder is located in the root of project.

By default, PlatformIO looks for boards in this order:

1. Project *boards_dir*
2. Global *home_dir*/boards
3. Development platform *home_dir*/platforms/*/boards.

This option can be overridden by global environment variable *PLATFORMIO_BOARDS_DIR*.

1.6.2 Section [*env:NAME*]

A section with *env:* prefix is used to define virtual environment with specific options that will be processed with *platformio run* command. You can define unlimited numbers of environments.

Each environment must have unique NAME. The valid chars for NAME are

- letters a–z
- numbers 0–9
- special char _ (underscore)

For example, [*env:hello_world*].

General options

- *platform*
- *framework*
- *board*
- *targets*

platform

Development Platforms name.

PlatformIO allows to use specific version of platform using [Semantic Versioning](#) (X.Y.Z=MAJOR.MINOR.PATCH) or VCS (Git, Mercurial and Subversion).

Version specifications can take any of the following forms:

- 0.1.2: an exact version number. Use only this exact version
- ^0.1.2: any compatible version (exact version for 0.x.x versions)
- ~0.1.2: any version with the same major and minor versions, and an equal or greater patch version
- >0.1.2: any version greater than 0.1.2. >=, <, and <= are also possible
- >0.1.0, !=0.2.0, <0.3.0: any version greater than 0.1.0, not equal to 0.2.0 and less than 0.3.0

Other forms are the same as for the *platformio platform install* command.

Examples:

```
[env:the_latest_version]
platform = atmelavr

[env:specific_major_version]
platform = atmelavr@^0.1.2

[env:specific_major_and_minor_version]
platform = atmelavr@~0.1.2

[env:development_verion_by_git]
platform = https://github.com/platformio/platform-ststm32.git

[env:custom_git_branch]
platform = https://github.com/platformio/platform-espressif8266.git#feature/stage

[env:specific_git_commit]
platform =
→https://github.com/platformio/platform-espressif8266.git#921855a9c530082efddb5d48b44c3f4be0e2dfa2
```

framework

Frameworks name.

The multiple frameworks are allowed, *split them with comma+space* “, “.

board

PlatformIO has pre-configured settings for the most popular boards. You don't need to specify `board_build.mcu`, `board_build.f_cpu`, `upload_protocol` or `upload_speed` options. Just define a board type and *PlatformIO* will pre-fill options described above with appropriate values.

You can find the board type in *Boards* section of each *Development Platforms* or using *PlatformIO Embedded Boards Explorer*.

targets

A list with targets which will be processed by `platformio run` command by default. You can enter more than one target, please split them with comma+space “, “.

The list with available targets is located in `platformio run --target`.

Example: build a project, upload firmware and start *Serial Monitor* automatically.

```
[env:upload_and_monitor]
targets = upload, monitor
```

Tip! You can use these targets like an option to `platformio run --target` command. For example:

```
# clean project
platformio run -t clean

# dump current build environment
platformio run --target envdump
```

When no targets are defined, *PlatformIO* will build only sources by default.

Board options

- `board_build.mcu`
- `board_build.f_cpu`
- [More options](#)

`board_build.mcu`

`board_build.mcu` is a microcontroller(MCU) type that is used by compiler to recognize MCU architecture. The correct type of `board_build.mcu` depends on platform library. For example, the list of `board_build.mcu` for “megaAVR Devices” is described [here](#).

The full list of `board_build.mcu` for the popular embedded platforms you can find in *Boards* section of *Development Platforms*. See “Microcontroller” column.

`board_build.f_cpu`

An option `board_build.f_cpu` is used to define MCU frequency (Hertz, Clock). A format of this option is C-like long integer value with L suffix. The 1 Hertz is equal to 1L, then 16 MHz (Mega Hertz) is equal to 16000000L.

The full list of `board_build.f_cpu` for the popular embedded platforms you can find in *Boards* section of *Development Platforms*. See “Frequency” column. You can overclock a board by specifying a `board_build.f_cpu` value other than the default.

More options

You can override any board option declared in manifest file using the next format `board_{OBJECT.PATH}`, where {OBJECT.PATH} is an object path in JSON manifest. Please navigate to “boards” folder of [PlatformIO development platforms](#) and open JSON file to list all available options.

For example, Manifest: Espressif ESP32 Dev Module:

```
[env:custom_board_options]
; Custom CPU Frequency
board_build.f_cpu = 160000000L

; Custom FLASH Frequency
board_build.f_flash = 80000000L

; Custom FLASH Mode
board_build.flash_mode = qio

; Custom maximum program size
board_upload.maximum_size = 1310720
```

Build options

- *build_flags*
 - *Built-in Variables*
 - *Dynamic build flags*
- *src_build_flags*
- *build_unflags*
- *src_filter*

build_flags

These flags/options control preprocessing, compilation, assembly and linking processes:

Format	Scope	Description
<code>-D name</code>	CPPDE-FINES	Predefine <i>name</i> as a macro, with definition 1.
<code>-D name=definition</code>	CPPDE-FINES	The contents of <i>definition</i> are tokenized and processed as if they appeared during translation phase three in a <code>#define</code> directive.
<code>-U name</code>	CPPDE-FINES	Cancel any previous definition of <i>name</i> , either built in or provided with a <code>-D</code> option.
<code>-Wp,option</code>	CPPFLAGS	Bypass the compiler driver and pass <i>option</i> directly through to the preprocessor
<code>-Wall</code>	CCFLAGS	Turns on all optional warnings which are desirable for normal code.
<code>-Werror</code>	CCFLAGS	Make all warnings into hard errors. Source code which triggers warnings will be rejected.
<code>-w</code>	CCFLAGS	Suppress all warnings, including those which GNU CPP issues by default.
<code>-include file</code>	CCFLAGS	Process <i>file</i> as if <code>#include "file"</code> appeared as the first line of the primary source file.
<code>-Idir</code>	CPPPATH	Add the directory <i>dir</i> to the list of directories to be searched for header files.
<code>-Wa,option</code>	ASFLAGS, CCFLAGS	Pass <i>option</i> as an option to the assembler. If <i>option</i> contains commas, it is split into multiple options at the commas.
<code>-Wl,option</code>	LINKFLAGS	Pass <i>option</i> as an option to the linker. If <i>option</i> contains commas, it is split into multiple options at the commas.
<code>-llibrary</code>	LIBS	Search the <i>library</i> named library when linking
<code>-Ldir</code>	LIBPATH	Add directory <i>dir</i> to the list of directories to be searched for <code>-l</code> .

This option can be set by global environment variable `PLATFORMIO_BUILD_FLAGS`.

For more detailed information about available flags/options go to:

- Options to Request or Suppress Warnings
- Options for Debugging Your Program
- Options That Control Optimization
- Options Controlling the Preprocessor
- Passing Options to the Assembler
- Options for Linking
- Options for Directory Search

Examples:

```
[env:specificDefines]
build_flags =
    -DFOO -DBAR=1
    -D BUILD_ENV_NAME=$PIOENV
    -D CURRENT_TIME=$UNIX_TIME
    -DFLOAT_VALUE=1.23457e+07

[env:stringDefines]
build_flags =
    -DHELLOW="World!"
    ; Password with special chars: My pass'word
    -DWIFI_PASS=\"My\ pass\'word\"

[env:specificInclibs]
build_flags =
    -I/opt/include
    -L/opt/lib
    -lfoo

[env:specificLdScript]
build_flags = -Wl,-T/path/to/ld_script.ld

[env:ignoreIncrementalBuilds]
; We dynamically change the value of "LAST_BUILD_TIME" macro,
; PlatformIO will not cache objects
build_flags = -DLAST_BUILD_TIME=$UNIX_TIME
```

Built-in Variables

You can inject into build flags built-in variables, such as:

- \$PYTHONEXE, full path to current Python interpreter
- \$UNIX_TIME, current time in Unix format
- \$PIOENV, name of build environment from *Project Configuration File platformio.ini*
- \$PIOPLATFORM, name of development platform
- \$PIOFRAMEWORK, name of framework
- \$PIOHOME_DIR, PlatformIO Home directory
- \$PROJECT_DIR, project directory
- \$PROJECTBUILD_DIR, project build directory per all environments
- \$BUILD_DIR, build directory per current environment
- Need more PlatformIO variables?

Please use target envdump for `platformio run --target` command to see ALL variables from a build environment.

Dynamic build flags

PlatformIO allows to run external command/script which outputs build flags into STDOUT. PlatformIO will automatically parse this output and append flags to a build environment.

You can use any shell or programming language.

This external command will be called on each *platformio run* command before building/uploading process.

Use Cases:

- Macro with the latest VCS revision/tag “on-the-fly”
- Generate dynamic headers (*.h)
- Process media content before generating SPIFFS image
- Make some changes to source code or related libraries

Note: If you need more advanced control and would like to apply changes to PlatformIO Build System environment, please refer to *Advanced Scripting*.

Example:

```
[env:generate_flags_with_external_command]
build_flags = !cmd_or_path_to_script

; Unix only, get output from internal command
build_flags = !echo "-DSOME_MACRO=$(some_cmd arg1 --option1)"
```

Use Case: Create “PIO_SRC_REV” macro with the latest Git revision

You will need to create a separate file named `git_rev_macro.py` and place it near `platformio.ini`.

`platformio.ini`:

```
[env:git_revision_macro]
build_flags = !python git_rev_macro.py
```

`git_rev_macro.py`:

```
import subprocess

revision = subprocess.check_output(["git", "rev-parse", "HEAD"]).strip()
print "-DPIO_SRC_REV=%s" % revision
```

`src_build_flags`

An option `src_build_flags` has the same behavior like `build_flags` but will be applied only for the project source code from `src_dir` directory.

This option can be set by global environment variable `PLATFORMIO_SRC_BUILD_FLAGS`.

`build_unflags`

Remove base/initial flags which were set by development platform.

```
[env:unflags]
build_unflags = -Os -std=gnu++11
build_flags = -O2
```

src_filter

This option allows to specify which source files should be included/excluded from build process. Filter supports 2 templates:

- +<PATH> include template
- -<PATH> exclude template

PATH MAST BE related from `src_dir`. All patterns will be applied in theirs order. GLOB Patterns are allowed.

By default, `src_filter` is predefined to `+<*> -<.git/> -<svn/> -<example/> -<examples/> -<test/> -<tests/>`, that means “includes ALL files, then exclude .git and svn repository folders, example... folder.

This option can be set by global environment variable `PLATFORMIO_SRC_FILTER`.

Upload options

- `upload_port`
- `upload_protocol`
- `upload_speed`
- `upload_flags`
- `upload_resetmethod`

upload_port

This option is used by “uploader” tool when sending firmware to board via `upload_port`. For example,

- /dev/ttyUSB0 - Serial port (Unix-based OS)
- COM3 - Serial port (Windows OS)
- 192.168.0.13 - IP address when using OTA
- /media/disk - physical path to media disk/flash drive (`mbed` enabled boards)
- D: - physical path to media disk/flash drive (Windows OS).

If `upload_port` isn't specified, then `PlatformIO` will try to detect it automatically.

To print all available serial ports use `platformio device list` command.

This option can be set by global environment variable `PLATFORMIO_UPLOAD_PORT`.

Please note that you can use Unix shell-style wildcards:

Pattern	Meaning
*	matches everything
?	matches any single character
[seq]	matches any character in seq
[!seq]	matches any character not in seq

Example

```
[env:uno]
platform = atmelavr
framework = arduino
; any port that starts with /dev/ttyUSB
upload_port = /dev/ttyUSB*

; COM1 or COM3
upload_port = COM[13]
```

upload_protocol

A protocol that “uploader” tool uses to talk to the board.

upload_speed

A connection speed ([baud rate](#)) which “uploader” tool uses when sending firmware to board.

upload_flags

Extra flags for uploader. Will be added to the end of uploader command. If you need to override uploader command or base flags please use [*extra_scripts*](#).

This option can be set by global environment variable [*PLATFORMIO_UPLOAD_FLAGS*](#).

Example

Please specify each flag/option in a new line starting with minimum 2 spaces.

```
[env:atmega328pb]
platform = atmelavr
board = atmega328pb
framework = arduino
upload_flags =
  -P$UPLOAD_PORT
  -b$UPLOAD_SPEED
  -u
  -Ulock:w:0xCF:m
  -Uhfuse:w:0xD7:m
  -Uefuse:w:0xF6:m
  -Ulfuse:w:0xE2:m
```

upload_resetmethod

Specify reset method for “uploader” tool. This option isn’t available for all development platforms. The only [Espressif 8266](#) supports it.

Monitor options

- *monitor_port*
- *monitor_speed*
- *monitor_rts*
- *monitor_dtr*

Custom options for [platformio device monitor](#) command.

monitor_port

Port, a number or a device name. See [platformio device monitor --port](#).

Example:

```
[env:custom_monitor_port]
...
; Unix
monitor_port = /dev/ttyUSB1

; Windows
monitor_port = COM3
```

monitor_speed

A monitor speed (baud rate). See [platformio device monitor --baud](#).

Example:

```
[env:custom_monitor_speedrate]
...
monitor_speed = 115200
```

monitor_rts

A monitor initial RTS line state. See [platformio device monitor --rts](#).

monitor_dtr

A monitor initial DTR line state. See [platformio device monitor --dtr](#).

Library options

New in version 3.0.

See also:

Please make sure to read [Library Dependency Finder \(LDF\)](#) guide first.

- `lib_deps`
- `lib_ignore`
- `lib_extra_dirs`
- `lib_ldf_mode`
- `lib_compat_mode`
- `lib_archive`

`lib_deps`

See also:

Please make sure to read [Library Dependency Finder \(LDF\)](#) guide first.

Specify project dependencies that should be installed automatically to `libdeps_dir` before environment processing. Multiple dependencies are allowed (*multi-lines or separated with comma+space* “, ”).

If you have multiple build environments that depend on the same libraries, you can use [Dynamic variables](#) to use common configuration.

Valid forms

```
; one line definition (comma + space)
[env:myenv]
lib_deps = LIBRARY_1, LIBRARY_2, LIBRARY_N

; multi-line definition
[env:myenv2]
lib_deps =
    LIBRARY_1
    LIBRARY_2
    LIBRARY_N
```

The each line with `LIBRARY_1`... `LIBRARY_N` will be passed automatically to `platformio lib install` command. Please follow to [platformio lib install](#) for detailed documentation about possible values.

Example:

```
[env:myenv]
lib_deps =
    13
    PubSubClient
    ArduinoJson@~5.6.!,=5.4
    https://github.com/gioblu/PJON.git#v2.0
    me-no-dev/ESPAsyncTCP
    IRremoteESP8266=https://github.com/markszabo/IRremoteESP8266/archive/master.zip
```

lib_ignore**See also:**

Please make sure to read [Library Dependency Finder \(LDF\)](#) guide first.

Specify libraries which should be ignored by Library Dependency Finder.

The correct value for this option is a library name (not folder name). You will see these names in “Library Dependency Graph” when building a project between < and > symbols.

The multiple library names are allowed, *split them with comma+space “, “* or put each library name in a separate new line.

Example:

Build output

```
...
Library Dependency Finder -> http://bit.ly/configure-pio-ldf
LDF MODES: FINDER(chain+) COMPATIBILITY(soft)
Collected 54 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <Hash> v1.0
|-- <AsyncMqttClient> v0.8.2
|   |-- <ESPAsyncTCP> v1.1.3
|-- <ESP8266WiFi> v1.0
|-- <ESP Async WebServer> v1.1.1
|   |-- <ESPAsyncTCP> v1.1.3
|   |-- <ESP8266WiFi> v1.0
|   |-- <Hash> v1.0
|   |-- <ArduinoJson> v5.13.1
|-- <ArduinoJson> v5.13.1
|-- <DNSServer> v1.1.0
|   |-- <ESP8266WiFi> v1.0
|-- <Ticker> v1.0
....
```

platformio.ini

```
[env:myenv]
; Single line
lib_ignore = AsyncMqttClient, DNSServer

; Multi-line
lib_ignore =
  AsyncMqttClient
  ESP Async WebServer
```

lib_extra_dirs**See also:**

Please make sure to read [Library Dependency Finder \(LDF\)](#) guide first.

A list with extra directories/storages where [Library Dependency Finder \(LDF\)](#) will look for dependencies. Multiple paths are allowed. *Please separate them using comma+space “, “*.

This option can be set by global environment variable `PLATFORMIO_LIB_EXTRA_DIRS` or using global [platformio] section and `lib_extra_dirs` option.

Warning: This is a not direct path to a library with source code. It should be a path to storage that contains libraries grouped by folders. For example, `D:\PlatformIO\extra\libraries` but not `D:\PlatformIO\extra\libraries\FooLibrary`.

Example:

```
[env:myenv]
lib_extra_dirs = /common/libraries, /iot/libraries
```

`lib_ldf_mode`

New in version 3.0.

See also:

Please make sure to read [Library Dependency Finder \(LDF\)](#) guide first.

This option specifies how does Library Dependency Finder should analyze dependencies (#include directives). See [Dependency Finder Mode](#) for details.

Example:

```
[env:myenv]
lib_ldf_mode = chain
```

`lib_compat_mode`

See also:

Please make sure to read [Library Dependency Finder \(LDF\)](#) guide first.

Library compatibility mode allows to control strictness of Library Dependency Finder. More details [Compatibility Mode](#).

By default, this value is set to `lib_compat_mode = soft` and means that LDF will check only for framework compatibility.

Example:

```
[env:myenv]
lib_compat_mode = soft
```

`lib_archive`

New in version 3.4.1.

Create an archive (*.a, static library) from the object files and link it into a firmware (program). This is default behavior of PlatformIO Build System (`lib_archive = true`).

Setting `lib_archive = false` will instruct PIO Build System to link object files directly (in-line). This could be useful if you need to override weak symbols defined in framework or other libraries.

You can disable library archiving per a custom library using `libArchive` field in `library.json` manifest.

Example:

```
[env:myenv]
lib_archive = false
```

Test options

New in version 3.0.

See also:

Please make sure to read [PIO Unit Testing](#) guide first.

- `test_filter`
- `test_ignore`
- `test_port`
- `test_speed`
- `test_transport`
 - *Examples*
- `test_build_project_src`

`test_filter`

Process only the [PIO Unit Testing](#) tests where the name matches specified patterns. Multiple names are allowed. Please separate them using comma+space “, “.

Also, you can filter some tests using `platformio test --filter` command.

Pattern	Meaning
*	matches everything
?	matches any single character
[seq]	matches any character in seq
[!seq]	matches any character not in seq

Example

```
[env:myenv]
test_filter = foottest, bartest_*, test[13]
```

`test_ignore`

Ignore [PIO Unit Testing](#) tests where the name matches specified patterns. Multiple names are allowed. Please separate them using comma+space “, “.

Also, you can ignore some tests using `platformio test --ignore` command.

Pattern	Meaning
*	matches everything
?	matches any single character
[seq]	matches any character in seq
[!seq]	matches any character not in seq

Example

```
[env:myenv]
test_ignore = foitest, bartest_*, test[13]
```

test_port

This option specifies communication interface (Serial/UART) between PlatformIO *PIO Unit Testing* Engine and target device. For example,

- /dev/ttyUSB0 - Unix-based OS
- COM3 - Windows OS

If `test_port` isn't specified, then *PlatformIO* will try to detect it automatically.

To print all available serial ports use `platformio device list` command.

test_speed

A connection speed ([baud rate](#)) to communicate with a target device. Default is 115200.

test_transport

PIO Unit Testing engine uses different transports to communicate with a target device. By default, it uses Serial/UART transport provided by a [framework](#). For example, when “`framework = arduino`”, the first available Serial will be used.

Default baudrate/speed is set to `test_speed`.

You can also define custom transport and implement its interface:

- `unittest_uart_begin();`
- `unittest_uart_putchar(char c);`
- `unittest_uart_flush();`
- `unittest_uart_end();`

Examples

1. Custom transport for *Native* platform

- Set `test_transport = custom` in *Project Configuration File* `platformio.ini`

```
[env:mycustomtransport]
platform = native
test_transport = custom
```

- Create `unittest_transport.h` file in project/test directory and implement prototypes above

```
#ifndef UNITTEST_TRANSPORT_H
#define UNITTEST_TRANSPORT_H

#include <stdio.h>

void unittest_uart_begin() {
}

void unittest_uart_putchar(char c) {
    putchar(c);
}

void unittest_uart_flush() {
    fflush(stdout);
}

void unittest_uart_end() {
}

#endif
```

2. STM32Cube HAL and Nucleo-F401RE: debugging and unit testing

`test_build_project_src`

Force *PIO Unit Testing* engine to build project source code from `src_dir` setting `test_build_project_src` to `true`. More detail about *Shared Code*.

Example

```
[env:myenv]
platform = ...
test_build_project_src = true
```

Debugging options

New in version 3.4.

See also:

Please make sure to read *PIO Unified Debugger* guide first.

- `debug_tool`
- `debug_init_break`

- `debug_init_cmds`
- `debug_extra_cmds`
- `debug_load_cmd`
- `debug_load_mode`
- `debug_server`
- `debug_port`
- `debug_svd_path`

`debug_tool`

A name of debugging tool. This option is useful when board supports more than one debugging tool (adapter, probe) or you want to create *Custom* debugging configuration.

See available tools in [Tools & Debug Probes](#).

Example

```
[env:debug]
platform = ...
board = ...
debug_tool = custom
```

`debug_init_break`

An initial breakpoint that makes your program stop whenever a certain point in the program is reached. **Default** value is set to `tbreak main` and means creating a temporary breakpoint at `int main(...)` function and automatically delete it after the first time a program stops there.

- [GDB Setting Breakpoints](#)
- [GDB Breakpoint Locations](#)

Note: Please note that each debugging tool (adapter, probe) has limited number of hardware breakpoints.

If you need more **Project Initial Breakpoints**, please place them in `debug_extra_cmds`.

Examples

```
[env:debug]
platform = ...
board = ...

; Examples 1: disable initial breakpoint
debug_init_break =

; Examples 2: temporary stop at ``void loop()`` function
debug_init_break = tbreak loop

; Examples 3: stop in main.cpp at line 13
debug_init_break = break main.cpp:13
```

`debug_init_cmds`

Initial commands that will be passed to back-end debugger.

PlatformIO dynamically configures back-end debugger depending on a debug environment. **Highly recommended to DO NOT override this option.**

For example, the custom initial commands for GDB:

```
[env:debug]
platform = ...
board = ...
debug_init_cmds =
    target extended-remote $DEBUG_PORT
    $INIT_BREAK
    monitor reset halt
    $LOAD_CMD
    monitor init
    monitor reset halt
```

`debug_extra_cmds`

Extra commands that will be passed to back-end debugger after initialization. For example, add custom breakpoint and load .gdbinit from a project directory for GDB:

```
[env:debug]
platform = ...
board = ...
debug_extra_cmds =
    break main.cpp:13
    break foo.cpp:100
    source .gdbinit
```

Note: Initial Project Breakpoints: Use `break path/to/file:LINE_NUMBER` to define initial breakpoints for debug environment. Multiple breakpoints are allowed.

To save session breakpoints, please use `save breakpoints [filename]` command in Debug Console. For example, `save breakpoints .gdbinit`. Later, this file could be loaded via `source [filename]` command. See above.

`debug_load_cmd`

Specify a command which will be used to load program/firmware to a target device. Possible options:

- `load - default` option
- some command - pass any debugging client command (GDB, etc.)
- `load address` - load program at specified address, where “address” should be a valid number
- `preload` - some embedded devices have locked Flash Memory (a few Freescale Kinetis and NXP LPC boards). In this case, firmware loading using debugging client is disabled. `preload` command instructs *PlatformIO Core* to load program/firmware using development platform “upload” method (via bootloader, media disk, etc)
- (empty value, `debug_load_cmd =`), disables program loading at all.

debug_load_mode

Allows to control when PlatformIO should load debugging firmware to the end target. Possible options:

- always - load for each debugging session, **default**
- modified - load only when firmware was modified
- manual - do not load firmware automatically. You are responsible to pre-flash target with debugging firmware in this case.

debug_server

Allows to setup a custom debugging server. By default, boards are pre-configured with a debugging server that is compatible with “on-board” debugging tool (adapter, probe). Also, this option is useful for a [Custom](#) debugging tool.

Option format (multi-line):

- First line is an executable path of debugging server
- 2-nd and the next lines are arguments for executable file

Example:

```
[env:debug]
platform = ...
board = ...
debug_server =
  /path/to/debugging/server
  arg1
  arg2
  ...
  argN
```

debug_port

A debugging port of a remote target. Could be a serial device or network address. PlatformIO detects it automatically if is not specified.

For example:

- /dev/ttyUSB0 - Unix-based OS
- COM3 - Windows OS
- localhost:3333

debug_svd_path

A custom path to [SVD file](#) which contains information about device peripherals.

Advanced options

New in version 3.4.1.

extra_scripts

See details and examples in [Advanced Scripting](#) section.

1.6.3 Dynamic variables

New in version 3.1.

Dynamic variables/templates are useful when you have common configuration data between build environments. For examples, common [build_flags](#) or project dependencies [lib_deps](#).

Each variable should have a next format: \${<section>.<option>}, where <section> is a value from [<section>] group, and <option> is a first item from pair <option> = value.

You can inject system environment variable using `sysenv` as a section. For example, \${`sysenv.HOME`}, etc.

- Variable can be applied only for the option's value
- Multiple variables are allowed
- The `platformio` and `env` sections are reserved and could not be used as a custom section. Some good section names might be `common` or `global`.

Example:

```
[platformio]
; Unix
lib_extra_dirs = ${sysenv.HOME}/Documents/Arduino/libraries
; Windows
lib_extra_dirs = ${sysenv.HOMEDRIVE}${sysenv.HOMEPATH}\Documents\Arduino\libraries

; You MUST inject these options into [env:] section
; using ${common.***} (see below)
[common]
build_flags = -D VERSION=1.2.3 -D DEBUG=1
lib_deps_builtin =
    SPI
    Wire
lib_deps_external = ArduinoJson@>5.6.0

[env:uno]
platform = atmelavr
framework = arduino
board = uno
build_flags = ${common.build_flags}
lib_deps =
    ${common.lib_deps_builtin}
    ${common.lib_deps_external}

[env:nodemcuv2]
platform = espressif8266
framework = arduino
board = nodemcuv2
build_flags = ${common.build_flags} -DSSID_NAME=HELLO -DSSID_PASSWORD=WORLD
lib_deps =
    ${common.lib_deps_builtin}
    ${common.lib_deps_external}
    PubSubClient@2.6
    OneWire
```

1.6.4 Examples

Note: A full list with project examples can be found in PlatformIO Repository.

1. *Atmel AVR*: Arduino UNO board with auto pre-configured `board_*` and `upload_*` options (use only `board` option) and Arduino Wiring-based Framework

```
[env:atmelavr_arduino_uno_board]
platform = atmelavr
framework = arduino
board = uno

; enable auto-uploading
targets = upload
```

2. *Atmel AVR*: Embedded board that is based on ATmega168 MCU with “arduino” bootloader

```
[env:atmelavr_atmega168_board]
platform = atmelavr
board_build.mcu = atmega168
board_build.f_cpu = 16000000L

upload_port = /dev/ttyUSB0
; for Windows OS
; upload_port = COM3
upload_protocol = arduino
upload_speed = 19200

; enable auto-uploading
targets = upload
```

3. Upload firmware via USB programmer (USBasp) to *Atmel AVR* microcontrollers

```
[env:atmelavr_usbasn]
platform = atmelavr
framework = arduino
board = pro8MHzatmega328
upload_protocol = usbasn
upload_flags = -Pusb -B5
```

Then upload firmware using target program for `platformio run --target`. command. To use other programmers see [Upload using Programmer](#).

4. *ST STM32*: Upload firmware using GDB script `upload.gdb`, issue #175

```
[env:st_via_gdb]
platform = ststm32
board = armstrap_eagle512
upload_protocol = gdb
```

Also, take a look at this article [Armstrap Eagle and PlatformIO](#).

5. *ST STM32*: Upload firmware using ST-Link instead mbed’s media disk

```
[env:stlink_for_mbed]
platform = ststm32
```

(continues on next page)

(continued from previous page)

```
board = disco_f100rb
upload_protocol = stlink
```

Example

```
[platformio]
env_default = nodemcuv2

; You MUST inject these options into [env:] section
; using ${common_env_data.***} (see below)
[common_env_data]
build_flags =
    -D VERSION=1.2.3
    -D DEBUG=1
lib_deps_builtin =
    SPI
    Wire
lib_deps_external =
    ArduinoJson@~5.6,!~5.4
    https://github.com/gioblu/PJON.git#v2.0
    IRremoteESP8266=https://github.com/markszabo/IRremoteESP8266/archive/master.zip

[env:nodemcuv2]
platform = espressif8266
framework = arduino
board = nodemcuv2

; Build options
build_flags =
    ${common_env_data.build_flags}
    -DSSID_NAME=HELLO
    -DSSID_PASSWORD=WORLD

; Library options
lib_deps =
    ${common_env_data.lib_deps_builtin}
    ${common_env_data.lib_deps_external}
    https://github.com/me-no-dev/ESPAsyncTCP.git
    PubSubClient@2.6
    OneWire

; Serial Monitor options
monitor_speed = 115200

; Unit Testing options
test_ignore = test_desktop

[env:bluepill_f103c8]
platform = ststm32
framework = arduino
board = bluepill_f103c8

; Build options
build_flags = ${common_env_data.build_flags}

; Library options
lib_deps =
```

(continues on next page)

(continued from previous page)

```
 ${common.lib_deps_external}

; Debug options
debug_tool = custom
debug_server =
    JLinkGDBServer
    -singlerun
    -if
    SWD
    -select
    USB
    -port
    2331
    -device
    STM32F103C8

; Unit Testing options
test_ignore = test_desktop
```

1.7 Environment variables

Environment variables are a set of dynamic named values that can affect the way running processes will behave on a computer.

PlatformIO handles variables which start with `PLATFORMIO_` prefix. They have the **HIGHEST PRIORITY**.

How to set environment variable?

```
# Windows
set VARIABLE_NAME=VALUE

# Unix (bash, zsh)
export VARIABLE_NAME=VALUE

# Unix (fish)
set -x VARIABLE_NAME VALUE
```

Contents

- [General](#)
- [Building](#)
- [Uploading](#)
- [Settings](#)

1.7.1 General

PlatformIO uses *General* environment variables for the common operations/commands.

CI

PlatformIO handles CI variable which is setup by Continuous Integration (Travis, Circle and etc.) systems. PlatformIO uses it to disable prompts and progress bars. In other words, CI=true automatically setup `PLATFORMIO_DISABLE_PROGRESSBAR` to true.

PLATFORMIO_AUTH_TOKEN

Allows to specify Personal Authentication Token that could be used for automatic login in to [PIO Account](#). It is very useful for [Continuous Integration](#) systems and [PIO Remote](#) operations where you are not able manually authorize.

You can get own Personal Authentication Token using `platformio account token` command.

PLATFORMIO_FORCE_COLOR

Force to output color ANSI-codes even if the output is a pipe (not a tty). The possible values are true and false. Default is `PLATFORMIO_FORCE_COLOR=false`.

PLATFORMIO_DISABLE_PROGRESSBAR

Disable progress bar for package/library downloader and uploader. This is useful when calling PlatformIO from subprocess and output is a pipe (not a tty). The possible values are true and false. Default is `PLATFORMIO_DISABLE_PROGRESSBAR=false`.

PLATFORMIO_HOME_DIR

Allows to override [Project Configuration File platformio.ini](#) option `home_dir`.

PLATFORMIO_INCLUDE_DIR

Allows to override [Project Configuration File platformio.ini](#) option `include_dir`.

PLATFORMIO_SRC_DIR

Allows to override [Project Configuration File platformio.ini](#) option `src_dir`.

PLATFORMIO_LIB_DIR

Allows to override [Project Configuration File platformio.ini](#) option `lib_dir`.

PLATFORMIO_LIBDEPS_DIR

Allows to override [Project Configuration File platformio.ini](#) option `libdeps_dir`.

PLATFORMIO_BUILD_DIR

Allows to override [Project Configuration File platformio.ini](#) option `build_dir`.

PLATFORMIO_DATA_DIR

Allows to override [Project Configuration File platformio.ini](#) option `data_dir`.

PLATFORMIO_TEST_DIR

Allows to override [Project Configuration File platformio.ini](#) option `test_dir`.

PLATFORMIO_BOARDS_DIR

Allows to override [Project Configuration File platformio.ini](#) option `boards_dir`.

PLATFORMIO_REMOTE_AGENT_DIR

Allows to override `platformio remote agent start --working-dir`.

1.7.2 Building

PLATFORMIO_BUILD_FLAGS

Allows to set *Project Configuration File platformio.ini* option `build_flags`.

Examples:

```
# Unix:  
export PLATFORMIO_BUILD_FLAGS==DFOO  
export PLATFORMIO_BUILD_FLAGS==DFOO -DBAR=1 -DFLOAT_VALUE=1.23457e+07  
export PLATFORMIO_BUILD_FLAGS='-DWIFI_PASS="My password"'  
→ '-DWIFI_SSID="My ssid name"'  
  
# Windows:  
SET PLATFORMIO_BUILD_FLAGS==DFOO  
SET PLATFORMIO_BUILD_FLAGS==DFOO -DBAR=1 -DFLOAT_VALUE=1.23457e+07  
SET PLATFORMIO_BUILD_FLAGS='-DWIFI_PASS="My password"' '-DWIFI_SSID="My ssid name"'
```

PLATFORMIO_SRC_BUILD_FLAGS

Allows to set *Project Configuration File platformio.ini* option `src_build_flags`.

PLATFORMIO_SRC_FILTER

Allows to set *Project Configuration File platformio.ini* option `src_filter`.

PLATFORMIO_EXTRA_SCRIPTS

Allows to set *Project Configuration File platformio.ini* option `extra_scripts`.

PLATFORMIO_LIB_EXTRA_DIRS

Allows to set *Project Configuration File platformio.ini* option `lib_extra_dirs`.

1.7.3 Uploading

PLATFORMIO_UPLOAD_PORT

Allows to set *Project Configuration File platformio.ini* option `upload_port`.

PLATFORMIO_UPLOAD_FLAGS

Allows to set *Project Configuration File platformio.ini* option `upload_flags`.

1.7.4 Settings

Allows to override PlatformIO settings. You can manage them via `platformio settings` command.

PLATFORMIO_SETTING_AUTO_UPDATE_LIBRARIES

Allows to override setting `auto_update_libraries`.

PLATFORMIO_SETTING_AUTO_UPDATE_PLATFORMS

Allows to override setting `auto_update_platforms`.

PLATFORMIO_SETTING_CHECK_LIBRARIES_INTERVAL

Allows to override setting `check_libraries_interval`.

PLATFORMIO_SETTING_CHECK_PLATFORMIO_INTERVAL

Allows to override setting `check_platformio_interval`.

PLATFORMIO_SETTING_CHECK_PLATFORMS_INTERVAL

Allows to override setting `check_platforms_interval`.

PLATFORMIO_SETTING_ENABLE_CACHE

Allows to override setting *enable_cache*.

PLATFORMIO_SETTING_ENABLE_SSL

Allows to override setting *enable_ssl*.

PLATFORMIO_SETTING_ENABLE_TELEMETRY

Allows to override setting *enable_telemetry*.

PLATFORMIO_SETTING_FORCE_VERBOSE

Allows to override setting *force_verbose*.

PLATFORMIO_SETTING_PROJECTS_DIR

Allows to override setting *projects_dir*.

1.8 Advanced Scripting

New in version 3.4.1.

Warning: Advanced Scripting is recommended for Advanced Users and requires Python language knowledge.

We highly recommend to take a look at *Dynamic build flags* option where you can use any programming language. Also, this option is useful if you need to apply changes to the project before building/uploading process:

- Macro with the latest VCS revision/tag “on-the-fly”
- Generate dynamic headers (*.h)
- Process media content before generating SPIFFS image
- Make some changes to source code or related libraries

More details *Dynamic build flags*.

Contents

- *Advanced Scripting*
 - *Launch types*
 - *Construction Environments*
 - *Before/Pre and After/Post actions*
 - *Examples*
 - * *Custom options in platformio.ini*
 - * *Split C/C++ build flags*
 - * *Extra Linker Flags without -Wl, prefix*
 - * *Custom upload tool*
 - * *Upload to Cloud (OTA)*
 - * *Custom firmware/program name*

* *Custom build target*

PlatformIO Build System allows to extend build process with the custom `extra_scripts` using Python interpreter and SCons construction tool. Build and upload flags, targets, toolchains data, and other information are stored in `SCons Construction Environments`.

Warning: You can not run/debug these scripts directly with Python interpreter. They will be loaded automatically when you processing project environment using `platformio run` command.

1.8.1 Launch types

There are 2 launch type of extra scripts:

1. **PRE** - executes before a main script of *Development Platforms*
2. **POST** - executes after a main script of *Development Platforms*

Multiple extra scripts are allowed. Please split them via “,” (comma + space) in the same line or use multi-line values.

For example, *Project Configuration File platformio.ini*

```
[env:my_env_1]
platform = ...
; without prefix, POST script
extra_scripts = post_extra_script.py

[env:my_env_2]
platform = ...
extra_scripts = pre:pre_extra_script1.py, pre:pre_extra_script2.py

[env:my_env_3]
platform = ...
extra_scripts =
  pre:pre_extra_script.py
  post:post_extra_script1.py
  post_extra_script2.py
```

This option can be set by global environment variable `PLATFORMIO_EXTRA_SCRIPTS`.

1.8.2 Construction Environments

There are 2 built-in construction environments which PlatformIO Build System uses to process a project:

- `env`, `Import ("env")` - global construction environment which is used for *Development Platforms* and *Frameworks* build scripts, upload tools, *Library Dependency Finder (LDF)*, and other internal operations
- `projenv`, `Import ("projenv")` - isolated construction environment which is used for processing of project source code located in `src_dir`. Please note that `src_build_flags` specified in *Project Configuration File platformio.ini* will be passed to `projenv` and not to `env`.

Note: `projenv` is available only for POST-type scripts

`extra_script.py`:

```

Import("env", "projenv")

# access to global construction environment
print env

# access to project construction environment
print projenv

#
# Dump construction environments (for debug purpose)
print env.Dump()
print projenv.Dump()

```

See examples below how to import construction environments and modify existing data or add new.

1.8.3 Before/Pre and After/Post actions

PlatformIO Build System has rich API that allows to attach different pre-/post actions (hooks) using `env.AddPreAction(target, callback)` or `env.AddPreAction(target, [callback1, callback2, ...])` function. A first argument `target` can be a name of target that is passed using `platformio run --target` command, a name of built-in targets (buildprog, size, upload, program, buildfs, uploadfs, uploadfsota) or path to file which PlatformIO processes (ELF, HEX, BIN, OBJ, etc.).

Examples

`extra_script.py` file is located on the same level as `platformio.ini`.

`platformio.ini`:

```

[env:pre_and_post_hooks]
extra_scripts = post:extra_script.py

```

`extra_script.py`:

```

Import("env", "projenv")

# access to global build environment
print env

# access to project build environment (is used source files in "src" folder)
print projenv

#
# Dump build environment (for debug purpose)
# print env.Dump()
#

#
# Change build flags in runtime
#
env.ProcessUnFlags("-DVECT_TAB_ADDR")
env.Append(CPPDEFINES=("VECT_TAB_ADDR", 0x123456789))

#
# Upload actions
#

```

(continues on next page)

(continued from previous page)

```

def before_upload(source, target, env):
    print "before_upload"
    # do some actions

    # call Node.JS or other script
    env.Execute("node --version")

def after_upload(source, target, env):
    print "after_upload"
    # do some actions

print "Current build targets", map(str, BUILD_TARGETS)

env.AddPreAction("upload", before_upload)
env.AddPostAction("upload", after_upload)

#
# Custom actions when building program/firmware
#

env.AddPreAction("buildprog", callback...)
env.AddPostAction("buildprog", callback...)

#
# Custom actions for specific files/objects
#

env.AddPreAction("${BUILD_DIR}/${PROGNAME}.elf", [callback1, callback2,...])
env.AddPostAction("${BUILD_DIR}/${PROGNAME}.hex", callback...)

# custom action before building SPIFFS image. For example, compress HTML, etc.
env.AddPreAction("${BUILD_DIR}/spiffs.bin", callback...)

# custom action for project's main.cpp
env.AddPostAction("${BUILD_DIR}/src/main.cpp.o", callback...)

# Custom HEX from ELF
env.AddPostAction(
    "${BUILD_DIR}/${PROGNAME}.elf",
    env.VerboseAction(" ".join([
        "$OBJCOPY", "-O", "ihex", "-R", ".eeprom",
        "${BUILD_DIR}/${PROGNAME}.elf", "${BUILD_DIR}/${PROGNAME}.hex"
    ]), "Building ${BUILD_DIR}/${PROGNAME}.hex")
)

```

1.8.4 Examples

Take a look at the multiple snippets/answers for the user questions:

- #365 Extra configuration for ESP8266 uploader
- #247 Specific options for avrdude.

Custom options in platformio.ini

platformio.ini:

```
[env:my_env]
platform = ...
extra_scripts = extra_script.py

custom_option1 = value1
custom_option2 = value2
```

extra_script.py:

```
from platformio import util

config = util.load_project_config()

value1 = config.get("my_env", "custom_option1")
value2 = config.get("my_env", "custom_option2")
```

Split C/C++ build flags

platformio.ini:

```
[env:my_env]
platform = ...
extra_scripts = extra_script.py
```

extra_script.py (place it near platformio.ini):

```
Import("env")

# General options that are passed to the C and C++ compilers
env.Append(CCFLAGS=["flag1", "flag2"])

# General options that are passed to the C compiler (C only; not C++).
env.Append(CFLAGS=["flag1", "flag2"])

# General options that are passed to the C++ compiler
env.Append(CXXFLAGS=["flag1", "flag2"])
```

Extra Linker Flags without -Wl, prefix

Sometimes you need to pass extra flags to GCC linker without `-Wl,`. You could use `build_flags` option but it will not work. PlatformIO will not parse these flags to `LINKFLAGS` scope. In this case, simple extra script will help:

platformio.ini:

```
[env:env_extra_link_flags]
platform = windows_x86
extra_scripts = extra_script.py
```

extra_script.py (place it near platformio.ini):

```
Import("env")

#
# Dump build environment (for debug)
# print env.Dump()
#

env.Append(
    LINKFLAGS=[
        "-static",
        "-static-libgcc",
        "-static-libstdc++"
    ]
)
```

Custom upload tool

You can override default upload command of development platform using extra script. There is the common environment variable UPLOADCMD which PlatformIO Build System will handle when you `platformio run -t upload`.

Please note that some development platforms can have more than 1 upload command. For example, `Atmel AVR` has UPLOADHEXCMD (firmware) and UPLOADEEPROMCMD (EEPROM data).

See examples below:

`platformio.ini`:

```
[env:my_custom_upload_tool]
platform = ...
# place it into the root of project or use full path
extra_scripts = extra_script.py
upload_protocol = custom
upload_flags = -arg1 -arg2 -argN
```

`extra_script.py` (place it near `platformio.ini`):

```
Import("env")

# please keep $SOURCE variable, it will be replaced with a path to firmware

# Generic
env.Replace(
    UPLOADER="executable or path to executable"
    UPLOADCMD="$UPLOADER $UPLOADERFLAGS $SOURCE"
)

# In-line command with arguments
env.Replace(
    UPLOADCMD="executable -arg1 -arg2 $SOURCE"
)

# Python callback
def on_upload(source, target, env):
    print source, target
    firmware_path = str(source[0])
    # do something
    env.Execute("executable arg1 arg2")
```

(continues on next page)

(continued from previous page)

```
env.Replace(UPLOADCMD=on_upload)
```

Upload to Cloud (OTA)

See project example <https://github.com/platformio/bintray-secure-ota>

Custom firmware/program name

Sometime is useful to have a different firmware/program name in *build_dir*.

platformio.ini:

```
[env:env_custom_prog_name]
platform = espressif8266
board = nodemcuv2
framework = arduino
build_flags = -D VERSION=13
extra_scripts = pre:extra_script.py
```

extra_script.py:

```
Import("env")

my_flags = env.ParseFlags(env['BUILD_FLAGS'])
defines = {k: v for (k, v) in my_flags.get("CPPDEFINES")}
# print defines

env.Replace(PROGNAME="firmware_%s" % defines.get("VERSION"))
```

Custom build target

There is a list with built-in targets which could be processed using `platformio run --target` option. You can create unlimited number of the own targets and declare custom handlers for them.

Let's create a simple ping target and process it with `platformio run --target ping` command:

platformio.ini:

```
[env:env_custom_target]
platform = ...
...
extra_scripts = extra_script.py
custom_ping_host = google.com
```

extra_script.py:

```
from platformio import util
from SCons.Script import AlwaysBuild
Import("env")

config = util.load_project_config()
host = config.get("env_custom_target", "custom_ping_host")
```

(continues on next page)

(continued from previous page)

```
def mytarget_callback(*args, **kwargs):
    print "Hello PlatformIO!"
    env.Execute("ping " + host)

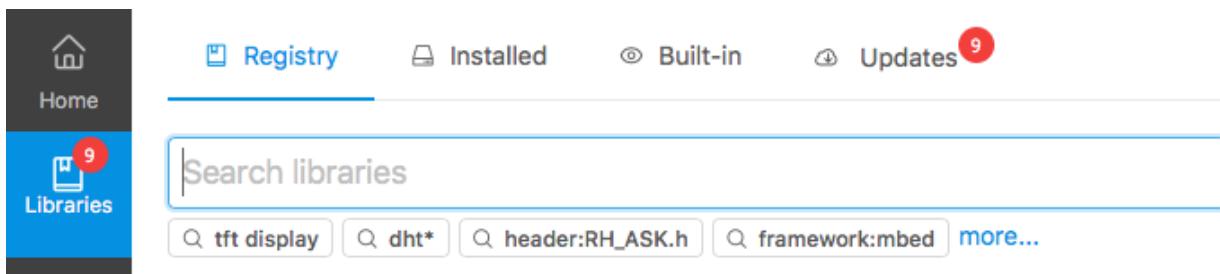
AlwaysBuild(env.Alias("ping", "", mytarget_callback))
```

1.9 Library Manager

PlatformIO Library Manager is a tool for managing libraries of [PlatformIO Registry](#) and VCS repositories (Git, Hg, SVN). It makes it exceedingly simple to find, install and keep libraries up-to-date. PlatformIO Library Manager supports [Semantic Versioning](#) and its rules.

[PlatformIO IDE](#) has built-in [PlatformIO Home](#) with a modern GUI which allows:

- Search for new libraries in PlatformIO Registry
- “1-click” library installation, per-project libraries, extra storages
- List installed libraries in multiple storages
- List built-in libraries (by frameworks)
- Updates for installed libraries
- Multiple examples, trending libraries, and more.



1.9.1 Quick Start

PlatformIO Library Manager is a tool for managing libraries of [PlatformIO Registry](#) and VCS repositories (Git, Hg, SVN). It makes it exceedingly simple to find, install and keep libraries up-to-date. PlatformIO Library Manager supports [Semantic Versioning](#) and its rules.

There are 3 options how to find/manage libraries:

- [PlatformIO Home](#)
- PIO Core [Command Line Interface](#)
- [Web Library Search](#)

You can manage different library storages using `platformio lib --global` or `platformio lib --storage-dir` options. If you change current working directory in terminal to project folder, then `platformio lib` command will manage automatically dependency storage in `libdeps_dir`.

Project dependencies

PlatformIO Library Manager allows to specify project dependencies (*lib_deps*) that will be installed automatically per project before environment processing. You do not need to install libraries manually. The only one simple step is to define dependencies in *Project Configuration File platformio.ini*. You can use library ID, Name or even repository URL. For example,

```
[env:myenv]
platform = ...
framework = ...
board = ...
lib_deps =
    13
    PubSubClient
    ArduinoJson@~5.6.!,!=5.4
    https://github.com/gioblu/PJON.git#v2.0
    https://github.com/me-no-dev/ESPAsyncTCP.git
    https://github.com/adafruit/DHT-sensor-library/archive/master.zip
```

Please follow to [platformio lib install](#) for detailed documentation about possible values.

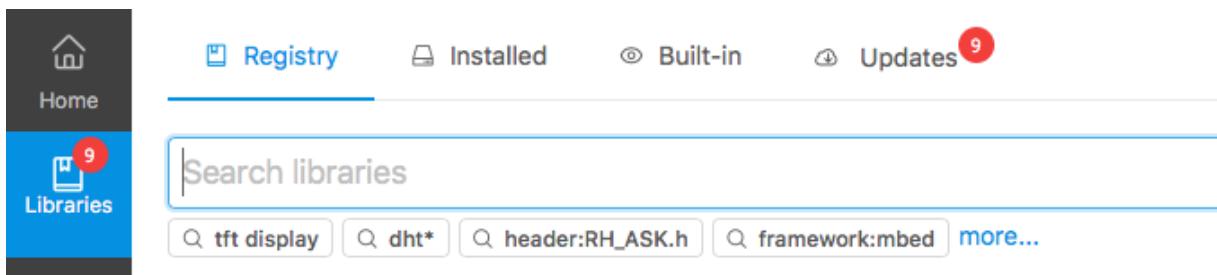
Warning: If some libraries are not visible in *PlatformIO IDE* and Code Completion or Code Linting does not work properly, please perform

- **Atom:** “Menu: PlatformIO > Rebuild C/C++ Project Index (Autocomplete, Linter)”
- **VSCode:** “Menu: View > Command Palette... > PlatformIO: Rebuild C/C++ Project Index”

PlatformIO IDE

PlatformIO IDE has built-in *PlatformIO Home* with a modern GUI which allows:

- Search for new libraries in PlatformIO Registry
- “1-click” library installation, per-project libraries, extra storages
- List installed libraries in multiple storages
- List built-in libraries (by frameworks)
- Updates for installed libraries
- Multiple examples, trending libraries, and more.



PlatformIO Core

1.9.2 Library Dependency Finder (LDF)

New in version 3.0.

Library Dependency Finder is a core part of PlatformIO Build System that operates with the C/C++ source files and looks for `#include ...` directives.

In spite of the fact that Library Dependency Finder is written in pure Python, it evaluates *C/C++ Preprocessor conditional syntax* (`#ifdef`, `if`, `defined`, `else`, and `elif`) without calling `gcc -E`. This approach allows significantly reduce total compilation time. See *Dependency Finder Mode* for more details.

Library Dependency Finder has controls that can be set up in *Project Configuration File* `platformio.ini`:

- `lib_deps`
 - `lib_extra_dirs`
 - `lib_ignore`
 - `lib_compat_mode`
 - `lib_ldf_mode`
 - `lib_archive`
-

Contents

- *Library Dependency Finder (LDF)*
 - *Storage*
 - *Dependency Finder Mode*
 - *Compatibility Mode*
 - *C/C++ Preprocessor conditional syntax*

Storage

There are different storages where Library Dependency Finder looks for libraries. These storages (folders) have priority and LDF operates in the next order:

1. `lib_extra_dirs` - extra storages per build environment
2. `lib_dir` - own/private library storage per project
3. `libdeps_dir` - project dependency storage used by *Library Manager*
4. “`home_dir/lib`” - global storage per all projects.
5. Library storages built into frameworks, SDKs.

Dependency Finder Mode

Library Dependency Finder starts work from analyzing source files of the project (`src_dir`) and can work in the next modes:

off “Manual mode”, does not process source files of a project and dependencies. Builds only the libraries that are specified in manifests (`library.json`, `module.json`) or using `lib_deps` option.

chain [DEFAULT] Parses ALL C/C++ source files of the project and follows only by nested includes (#include . . . , chain...) from the libraries. It also parses C, CC, CPP files from libraries which have the same name as included header file. **Does not evaluates C/C++ Preprocessor conditional syntax.**

deep Parses ALL C/C++ source files of the project and parses ALL C/C++ source files of the each found dependency (recursively). **Does not evaluates C/C++ Preprocessor conditional syntax.**

chain+ The same behavior as for the `chain` but **evaluates C/C++ Preprocessor conditional syntax**.

deep+ The same behavior as for the `deep` but **evaluates C/C++ Preprocessor conditional syntax**.

The mode can be changed using `lib_ldf_mode` option in *Project Configuration File platformio.ini*. Default value is set to `chain`.

Note: Usually, when the LDF appears to fail to identify a dependency of a library, it is because the dependency is only referenced from the library source file, and not the library header file (see example below). In this case, it is necessary to either explicitly reference the dependency from the project source or *Project Configuration File platformio.ini* (`lib_deps` option), or change the LDF mode to “deep” (not generally recommended).

A difference between `chain/chain+` and `deep/deep+` modes. For example, there are 2 libraries:

- Library “Foo” with files:
 - Foo/foo.h
 - Foo/foo.cpp
 - Foo/extracpp
- Library “Bar” with files:
 - Bar/bar.h
 - Bar/bar.cpp

Case 1

- `lib_ldf_mode = chain`
- `Foo/foo.h` depends on “Bar” library (contains `#include <bar.h>`)
- `#include <foo.h>` is located in one of the project source files

Here are nested includes (project file > `foo.h` > `bar.h`) and LDF will find both libraries “Foo” and “Bar”.

Case 2

- `lib_ldf_mode = chain`
- `Foo/extracpp` depends on “Bar” library (contains `#include <bar.h>`)
- `#include <foo.h>` is located in one of the project source files

In this case, LDF will not find “Bar” library because it doesn’t know about CPP file (`Foo/extracpp`).

Case 3

- `lib_ldf_mode = deep`
- `Foo/extracpp` depends on “Bar” library (contains `#include <bar.h>`)
- `#include <foo.h>` is located in one of the project source files

Firstly, LDF finds “Foo” library, then it parses all sources from “Foo” library and finds `Foo/extracpp` that depends on `#include <bar.h>`. Secondly, it will parse all sources from “Bar” library and this operation continues until all dependencies will not be parsed.

Compatibility Mode

Compatibility mode allows to control strictness of Library Dependency Finder. If library contains one of manifest file (`library.json`, `library.properties`, `module.json`), then LDF check compatibility of this library with real build environment. Available compatibility modes:

off Does not check for compatibility (is not recommended)

soft [DEFAULT] Checks for the compatibility with `framework` from build environment

strict Checks for the compatibility with `framework` and `platform` from build environment.

This mode can be changed using `lib_compat_mode` option in *Project Configuration File platformio.ini*. Default value is set to `soft`.

C/C++ Preprocessor conditional syntax

In spite of the fact that Library Dependency Finder is written in pure Python, it evaluates C/C++ Preprocessor conditional syntax (`#ifdef`, `if`, `defined`, `else`, and `elif`) without calling `gcc -E`. For example,

`platformio.ini`

```
[env:myenv]
lib_ldf_mode = chain+
build_flags = -D MY_PROJECT_VERSION=13
```

`mylib.h`

```
#ifdef PROJECT_VERSION
// include common file for the project
#include "my_common.h"
#endif

#if PROJECT_VERSION < 10
// this include will be ignored because does not satisfy condition above
#include "my_old.h"
#endif
```

1.9.3 `library.json`

`library.json` is a manifest file of development library. It allows developers to keep project in own structure and define:

- location of source code
- examples list
- compatible frameworks and platforms
- library dependencies
- advanced build settings

PlatformIO Library Crawler uses `library.json` manifest to extract source code from developer's location and keeps a cleaned library in own Library Registry.

A data in `library.json` should be represented in JSON-style via associative array (name/value pairs). An order doesn't matter. The allowable fields (names from pairs) are described below.

Fields

- `name`
- `description`
- `keywords`
- `authors`
- `repository`
- `version`
- `license`
- `downloadUrl`
- `homepage`
- `export`
 - `include`
 - `exclude`
- `frameworks`
- `platforms`
- `dependencies`
- `examples`
- `build`
 - `flags`
 - `unflags`
 - `srcFilter`
 - `extraScript`
 - `libArchive`
 - `libLDFMode`
 - `libCompatMode`
- `Examples`

`name`

Required | Type: String | Max. Length: 50

A name of the library.

- Must be unique.

- Should be slug style for simplicity, consistency and compatibility. Example: *Arduino-SPI*
- Title Case, Aa-z, can contain digits and dashes (but not start/end with them).
- Consecutive dashes are not allowed.

description

Required | Type: String | Max. Length: 255

The field helps users to identify and search for your library with a brief description. Describe the hardware devices (sensors, boards and etc.) which are suitable with it.

keywords

Required | Type: String | Max. Length: 255

Used for search by keyword. Helps to make your library easier to discover without people needing to know its name.

The keyword should be lowercased, can contain a-z, digits and dash (but not start/end with them). A list from the keywords can be specified with separator ,

authors

Required if *repository* field is not defined | Type: Object or Array

An author contact information

- name Full name (**Required**)
- email
- url An author's contact page
- maintainer Specify “maintainer” status

Examples:

```
"authors":  
{  
    "name": "John Smith",  
    "email": "me@john-smith.com",  
    "url": "http://www.john-smith/contact"  
}  
  
...  
  
"authors":  
[  
    {  
        "name": "John Smith",  
        "email": "me@john-smith.com",  
        "url": "http://www.john-smith/contact"  
    },  
    {  
        "name": "Andrew Smith",  
        "email": "me@andrew-smith.com",  
        "url": "http://www.andrew-smith/contact",  
    }  
]
```

(continues on next page)

(continued from previous page)

```

        "maintainer": true
    }
]
```

Note: You can omit `authors` field and define `repository` field. Only *GitHub-based* repository is supported now. In this case *PlatformIO Library Registry Crawler* will use information from [GitHub API Users](#).

repository

Required if `downloadUrl` field is not defined | Type: Object

The repository in which the source code can be found. The field consists of the next items:

- type the only “git”, “hg” or “svn” are supported
- url
- branch if is not specified, default branch will be used. This field will be ignored if tag/release exists with the value of `version`.

Example:

```

"repository":
{
    "type": "git",
    "url": "https://github.com/foo/bar.git"
}
```

version

Required if `repository` field is not defined | Type: String | Max. Length: 20

A version of the current library source code. Can contain a-z, digits, dots or dash. [Semantic Versioning](#) IS RECOMMENDED.

Case 1 `version` and `repository` fields are defined. The `repository` is hosted on GitHub or Bitbucket.

PlatformIO Library Registry Crawler will lookup for release tag named as value of `version` or with v prefix (you do not need to pass this v prefix to the `version` field).

Case 2 `version` and `repository` fields are defined and `repository` does not contain tag/release with value of `version`.

PlatformIO Library Registry Crawler will use the latest source code from `repository` and link it with specified `version`. If `repository` branch is not specified, then default branch will be used. Also, if you push new commits to `repository` and do not update `version` field, the library will not be updated until you change the `version`.

Case 3 `version` field is not defined and `repository` field is defined.

PlatformIO Library Registry Crawler will use the [VCS](#) revision from the latest commit as “current version”. For example, 1.3 (SVN) or first 10 chars of SHA digest e4564b7da4 (Git). If `repository` branch is not specified, then default branch will be used.

We recommend to use `version` field and specify the real release version and make appropriate tag in the `repository`. In other case, users will receive updates for library with each new commit to `repository`.

Note:

PlatformIO Library Registry Crawler updates library only if:

- the `version` is changed
 - `library.json` is modified
-

Example:

```
"repository":  
{  
    "type": "git",  
    "url": "https://github.com/foo/bar.git"  
},  
"version": "1.0.0"
```

license

Optional | Type: String

A license of the library. You can check the full list of SPDX license IDs. Ideally you should pick one that is OSI approved.

```
"license": "Apache-2.0"
```

downloadUrl

Required if repository field is not defined | Type: String

It is the *HTTP URL* to the archived source code of library. It should end with the type of archive (`.zip` or `.tar.gz`).

Note: `downloadUrl` has higher priority than `repository`.

Example with fixed release/tag on GitHub:

```
"version": "1.0.0",  
"downloadUrl": "https://github.com/foo/bar/archive/v1.0.0.tar.gz",  
"include": "bar-1.0.0"
```

See more `library.json` *Examples*.

homepage

Optional | Type: String | Max. Length: 255

Home page of library (if is different from `repository` url).

export*Optional* | Type: Object

Explain PlatformIO Library Crawler which content from the repository/archive should be exported as “source code” of the library. This option is useful if need to exclude extra data (test code, docs, images, PDFs, etc). It allows to reduce size of the final archive.

Possible options:

- *include*
- *exclude*

include*Optional* | Type: String or Array | Glob Pattern

If `include` field is a type of String, then *PlatformIO Library Registry Crawler* will recognize it like a “relative path inside repository/archive to library source code”. See example below where the only source code from the relative directory `LibrarySourceCodeHere` will be included.

```
"include": "some/child/dir/LibrarySourceCodeHere"
```

If `include` field is a type of Array, then *PlatformIO Library Registry Crawler* firstly will apply `exclude` filter and then include only directories/files which match with `include` patterns.

Example:

```
"export": {
    "include": [
        "dir/*.cpp",
        "dir/examples/*",
        "*/*/*.h"
    ]
}
```

Pattern Meaning

Pattern	Meaning
*	matches everything
?	matches any single character
[seq]	matches any character in seq
[!seq]	matches any character not in seq

See more `library.json` *Examples*.

exclude*Optional* | Type: String or Array | Glob Pattern

Exclude the directories and files which match with `exclude` patterns.

frameworks

Optional | Type: String or Array

A list with compatible frameworks. The available framework types are defined in the [Development Platforms](#) section.
If the library is compatible with the all frameworks, then you can use * symbol:

```
"frameworks": "*"
```

platforms

Optional | Type: String or Array

A list with compatible platforms. The available platform types are defined in [Development Platforms](#) section.
If the library is compatible with the all platforms, then you can use * symbol:

```
"platforms": "*"
```

dependencies

Optional | Type: Array or Object

A list of dependent libraries. They will be installed automatically with [platformio lib install](#) command.

Allowed requirements for dependent library:

- name | Type: String
- version | Type: String
- authors | Type: String or Array
- frameworks | Type: String or Array
- platforms | Type: String or Array

The version supports Semantic Versioning (<major>.<minor>.<patch>) and can take any of the following forms:

- 0.1.2 - an exact version number. Use only this exact version
- ^0.1.2 - any compatible version (exact version for 0.x.x versions)
- ~0.1.2 - any version with the same major and minor versions, and an equal or greater patch version
- >0.1.2 - any version greater than 0.1.2. >=, <, and <= are also possible
- >0.1.0, !=0.2.0, <0.3.0 - any version greater than 0.1.0, not equal to 0.2.0 and less than 0.3.0

The rest possible values including VCS repository URLs are documented in [platformio lib install](#) command.

Example:

```
"dependencies":  
[  
  {  
    "name": "Library-Foo",  
    "authors":  
    [  
      {  
        "name": "John Doe",  
        "email": "john.doe@example.com"  
      },  
      {  
        "name": "Jane Doe",  
        "email": "jane.doe@example.com"  
      }  
    ]  
  }  
]
```

(continues on next page)

(continued from previous page)

```

        "Jhon Smith",
        "Andrew Smith"
    ],
},
{
    "name": "Library-Bar",
    "version": "~1.2.3"
},
{
    "name": "lib-from-repo",
    "version": "https://github.com/user/package.git#1.2.3"
}
]

```

A short definition of dependencies is allowed:

```

"dependencies": {
    "mylib": "1.2.3",
    "lib-from-repo": "githubuser/package"
}

```

See more `library.json` *Examples*.

examples

Optional | Type: String or Array | Glob Pattern

A list of example patterns. This field is predefined with default value:

```

"examples": [
    "[Ee]xamples/*.c",
    "[Ee]xamples/*.cpp",
    "[Ee]xamples/*.ino",
    "[Ee]xamples/*.pde",
    "[Ee]xamples/*/*.c",
    "[Ee]xamples/*/*.cpp",
    "[Ee]xamples/*/*.ino",
    "[Ee]xamples/*/*.pde",
    "[Ee]xamples/*/*/*.c",
    "[Ee]xamples/*/*/*.cpp",
    "[Ee]xamples/*/*/*.ino",
    "[Ee]xamples/*/*/*.pde"
]

```

build

Optional | Type: Object

Specify advanced settings, options and flags for the build system. Possible options:

- *flags*
- *unflags*

- `srcFilter`
- `extraScript`
- `libArchive`
- `libLDFMode`
- `libCompatMode`

`flags`

Optional | Type: String or Array

Extra flags to control preprocessing, compilation, assembly and linking processes. More details [build_flags](#).

`unflags`

Optional | Type: String or Array

Remove base/initial flags which were set by development platform. More details [build_unflags](#).

`srcFilter`

Optional | Type: String or Array

Specify which source files should be included/excluded from build process. The path in filter should be **relative from a root of library**.

See syntax in [src_filter](#).

Please note that you can generate source filter “on-the-fly” using `extraScript` (see below)

`extraScript`

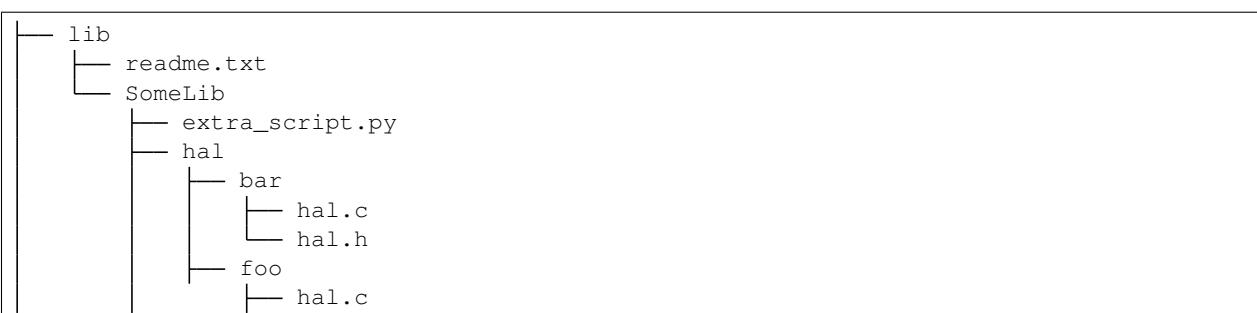
Optional | Type: String

Launch extra script before build process. More details [extra_scripts](#).

Example (HAL-based library)

This example demonstrates how to build HAL-dependent source files and exclude other source files from a build process.

Project structure



(continues on next page)

(continued from previous page)



platformio.ini

```

[env:foo]
platform = native
build_flags = -DHAL=foo

[env:bar]
platform = native
build_flags = -DHAL=bar

```

library.json

```
{
    "name": "SomeLib",
    "version": "0.0.0",
    "build": {
        "extraScript": "extra_script.py"
    }
}
```

extra_script.py

```

Import('env')
from os.path import join, realpath

for item in env.get("CPPDEFINES", []):
    if isinstance(item, tuple) and item[0] == "HAL":
        env.Append(CPPPATH=[realpath(join("hal", item[1]))])
        env.Replace(SRC_FILTER=[ "+<*>", "-<hal>", "+<%s>" % join("hal", item[1]) ])
        break

```

libArchive

Optional | Type: Boolean

Create an archive (*.a, static library) from the object files and link it into a firmware (program). This is default behavior of PlatformIO Build System ("libArchive": true).

Setting "libArchive": false will instruct PIO Build System to link object files directly (in-line). This could be useful if you need to override weak symbols defined in framework or other libraries.

You can disable library archiving globally using `lib_archive` option in *Project Configuration File* `platformio.ini`.

libLDFMode

Optional | Type: String

Specify Library Dependency Finder Mode. See [Dependency Finder Mode](#) for details.

`libCompatMode`

Optional | Type: Integer

Specify Library Compatibility Mode. See [Compatibility Mode](#) for details.

Examples

1. Custom macros/defines

```
"build": {
    "flags": "-D MYLIB_REV=0.1.2 -DRELEASE"
}
```

2. Extra includes for C preprocessor

```
"build": {
    "flags": [
        "-I inc",
        "-I inc/target_x13"
    ]
}
```

3. Force to use C99 standard instead of C11

```
"build": {
    "unflags": "-std=gnu++11",
    "flags": "-std=c99"
}
```

4. Build source files (c, cpp, h) at the top level of the library

```
"build": {
    "srcFilter": [
        "+<*.c>",
        "+<*.cpp>",
        "+<*.h>"
    ]
}
```

5. Extend PlatformIO Build System with own extra script

```
"build": {
    "extraScript": "generate_headers.py"
}
```

`generate_headers.py`

```
Import('env')
# print env.Dump()
env.Append(
    CPPDEFINES=["HELLO=WORLD", "TAG=1.2.3", "DEBUG"],
    CPPPATH=["inc", "inc/devices"]
)
```

(continues on next page)

(continued from previous page)

```
# some python code that generates header files "on-the-fly"
```

1.9.4 Creating Library

PlatformIO Library Manager doesn't have any requirements to a library source code structure. The only one requirement is library's manifest file - `library.json`. It can be located inside your library or in the another location where *PlatformIO Library Registry Crawler* will have *HTTP* access.

Updates to existing libraries are done every 24 hours. In case a more urgent update is required, you can post a request on PlatformIO [community](#).

Contents

- *Source Code Location*
 - *At GitHub*
 - *Under VCS (SVN/GIT)*
 - *Self-hosted*
- *Register*
- *Examples*

Source Code Location

There are a several ways how to share your library with the whole world (see [examples](#)).

You can hold a lot of libraries (split into separated folders) inside one of the repository/archive. In this case, you need to specify `include` option of `export` field to relative path to your library's source code.

At GitHub

Recommended

If a library source code is located at [GitHub](#), then you **need to specify** only these fields in the `library.json`:

- `name`
- `version` (is not required, but highly recommended for new *Library Manager*)
- `keywords`
- `description`
- `repository`

PlatformIO Library Registry Crawler will populate the rest fields, like `authors` with an actual information from [GitHub](#).

Example, [DallasTemperature](#):

```
{  
    "name": "DallasTemperature",  
    "keywords": "onewire, 1-wire, bus, sensor, temperature",  
    "description": "  
        → Arduino Library for Dallas Temperature ICs (DS18B20, DS18S20, DS1822, DS1820)",  
    "repository":  
    {  
        "type": "git",  
        "url": "https://github.com/milesburton/Arduino-Temperature-Control-Library.git"  
    },  
    "authors":  
    [  
        {  
            "name": "Miles Burton",  
            "email": "miles@mnetcs.com",  
            "url": "http://www.milesburton.com",  
            "maintainer": true  
        },  
        {  
            "name": "Tim Newsome",  
            "email": "nuisance@casualhacker.net"  
        },  
        {  
            "name": "Guil Barros",  
            "email": "gfbbarros@bappos.com"  
        },  
        {  
            "name": "Rob Tillaart",  
            "email": "rob.tillaart@gmail.com"  
        }  
    ],  
    "dependencies":  
    {  
        "name": "OneWire",  
        "authors": "Paul Stoffregen",  
        "frameworks": "arduino"  
    },  
    "version": "3.7.7",  
    "frameworks": "arduino",  
    "platforms": "*"  
}
```

Under VCS (SVN/GIT)

PlatformIO Library Registry Crawler can operate with a library source code that is under *VCS* control. The list of **required** fields in the `library.json` will look like:

- `name`
- `keywords`
- `description`
- `authors`
- `repository`

Example:

```
{
  "name": "XBee",
  "keywords": "xbee, protocol, radio",
  "description": "Arduino library for communicating with XBee in API mode",
  "authors": [
    {
      "name": "Andrew Rapp",
      "email": "andrew.rapp@gmail.com",
      "url": "https://code.google.com/u/andrew.rapp@gmail.com/"
    },
    "repository": [
      {
        "type": "git",
        "url": "https://code.google.com/p/xbee-arduino/"
      }
    ],
    "frameworks": "arduino",
    "platforms": "atmelavr"
  }
}
```

Self-hosted

You can manually archive (*Zip*, *Tar.Gz*) your library source code and host it in the *Internet*. Then you should specify the additional fields, like *version* and *downloadUrl*. The final list of **required** fields in the *library.json* will look like:

- *name*
 - *keywords*
 - *description*
 - *authors*
 - *version*
 - *downloadUrl*

```
{  
    "name": "OneWire",  
    "keywords": "onewire, 1-wire, bus, sensor, temperature, ibutton",  
    "description": _  
→ "Control devices (from Dallas Semiconductor) that use the One Wire protocol (DS18S20, DS18B20, DS24...  
→  
    "authors":  
    {  
        "name": "Paul Stoffregen",  
        "url": "http://www.pjrc.com/teensy/td_libs_OneWire.html"  
    },  
    "version": "2.2",  
    "downloadUrl": "http://www.pjrc.com/teensy/arduino_libraries/OneWire.zip",  
    "export": {  
        "include": "OneWire"  
    },  
    "frameworks": "arduino",  
    "platforms": "atmelavr"  
}
```

Register

The registration requirements:

- A library must adhere to the *library.json* specification.
- There must be public *HTTP* access to the library *library.json* file.

Now, you can *register* your library and allow others to *install* it.

Examples

Command:

```
$ platformio lib register http://my.example.com/library.json
```

- GitHub + fixed release
- Dependencies by author and framework
- Multiple libraries in the one repository

1.10 Development Platforms

PlatformIO ecosystem has decentralized architecture. Build scripts, toolchains, the pre-built tools for the popular OS (*Mac OS X, Linux (+ARM) and Windows*) are organized into the multiple development platforms.

Each development platform contains:

- **PlatformIO Build System** based build scripts for the supported frameworks and SDKs
- Pre-configured presets for embedded boards
- Pre-compiled toolchains and relative tools for multiple architectures.

A platform name or its specific version could be specified using *platform* option in *Project Configuration File platformio.ini*.

1.10.1 Embedded

Atmel AVR

platform = atmelavr

Atmel AVR 8- and 32-bit MCUs deliver a unique combination of performance, power efficiency and design flexibility. Optimized to speed time to market-and easily adapt to new ones-they are based on the industry's most code-efficient architecture for C and assembly programming.

For more detailed information please visit [vendor site](#).

Contents

- *Configuration*
- *Examples*

- *Stable and upstream versions*
- *Packages*
- *Frameworks*
- *Boards*

Configuration

- *Upload using Programmer*
- *Upload EEPROM data*
- *Fuses*
 - *Custom Fuses*

Upload using Programmer

To upload firmware using programmer you need to use program target instead of upload for `platformio run --target` command. For example, `platformio run -t program`.

Warning: Upload options like `upload_port` don't work as expected with `platformio run -t program`. You need to use `upload_flags` if you want to specify custom port or speed (see examples below).

Note: List of avrdude supported programmers are accessible with `avrdude -c ?`

Configuration for the programmers:

- AVR ISP

```
[env:myenv]
platform = atmelavr
framework = arduino
upload_protocol = stk500v1
upload_flags = -P$UPLOAD_PORT

; edit this line with valid upload port
upload_port = SERIAL_PORT_HERE
```

- AVRISP mkII

```
[env:myenv]
platform = atmelavr
framework = arduino
upload_protocol = stk500v2
upload_flags = -Pusb
```

- USBtinyISP

```
[env:myenv]
platform = atmelavr
framework = arduino
upload_protocol = usbtiny
```

- ArduinoISP

```
[env:myenv]
platform = atmelavr
framework = arduino
upload_protocol = arduinoisp
```

- USBasp

```
[env:myenv]
platform = atmelavr
framework = arduino
upload_protocol = usbasp
upload_flags = -Pusb
```

- Parallel Programmer

```
[env:myenv]
platform = atmelavr
framework = arduino
upload_protocol = dapa
upload_flags = -F
```

- Arduino as ISP

```
[env:myenv]
platform = atmelavr
framework = arduino
upload_protocol = stk500v1
; each flag in a new line
upload_flags =
    -P$UPLOAD_PORT
    -b$UPLOAD_SPEED

; edit these lines
upload_port = SERIAL_PORT_HERE
upload_speed = 19200
```

- Bus Pirate as ISP

```
[env:myenv]
platform = atmelavr
framework = arduino
upload_protocol = buspirate
; each flag in a new line
upload_flags =
    -P$UPLOAD_PORT
    -b$UPLOAD_SPEED

; edit these lines
upload_port = SERIAL_PORT_HERE
upload_speed = 115200
```

Upload EEPROM data

To upload EEPROM data (from EEMEM directive) you need to use `uploaddeep` target instead `upload` for `platformio run --target` command. For example, `platformio run -t uploaddeep`.

Fuses

PlatformIO has built-in target named `fuses` for setting fuse bits. The default fuse bits are predefined in board manifest file in `fuses` section. For example, [Arduino Uno Fuses](#).

To set fuse bits you need to use target `fuses` for `platformio run --target` command.

Custom Fuses

You can specify custom fuse bits. Please create custom `extra_scripts` and override default “fuses” command:

`platformio.ini`:

```
[env:custom_fuses]
platform = atmelavr
extra_scripts = extra_script.py
```

`extra_script.py`:

```
Import('env')
env.
    Replace(FUSESCMD="avrdude $UPLOADERFLAGS -e -Ulock:w:0x3F:m -Uhfuse:w:0xDE:m -Uefuse:w:0x05:m -Ulfu
```

Examples

Examples are listed from Atmel AVR development platform repository:

- arduino-blink
- arduino-external-libs
- arduino-internal-libs
- arduino-own-src_dir
- digitstump-mouse
- engduino-magnetometer
- native-blink
- simba-blink

Stable and upstream versions

You can switch between stable releases of Atmel AVR development platform and the latest upstream version using `platform` option in `Project Configuration File platformio.ini` as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = atmelavr
board = ...

; Custom stable version
[env:custom_stable]
platform = atmelavr@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-atmelavr.git
board = ...
```

Packages

Name	Description
framework-arduinoavr	Arduino Wiring-based Framework (AVR Core, 1.6)
framework-simba	Simba Framework
tool-avrdude	AVRDUDE
tool-micronucleus	Micronucleus
toolchain-atmelavr	avr-gcc

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Ar-duino	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.
Simba	Simba is an RTOS and build framework. It aims to make embedded programming easy and portable.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

Adafruit

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
bluefruitmicro	Adafruit Bluefruit Micro	No	AT-MEGA32U4	8MHz	28KB	2.50KB
feather328p	Adafruit Feather 328P	No	AT-MEGA328P	8MHz	31.50KB	2KB
feather32u4	Adafruit Feather 32u4	No	AT-MEGA32U4	8MHz	28KB	2.50KB
flora8	Adafruit Flora	No	AT-MEGA32U4	8MHz	28KB	2.50KB
gemma	Adafruit Gemma	No	ATTINY85	8MHz	8KB	512B
itsybitsy32u4_3	Adafruit ItsyBitsy 3V/8MHz	No	AT-MEGA32U4	8MHz	28KB	2.50KB
itsybitsy32u4_5	Adafruit ItsyBitsy 5V/16MHz	No	AT-MEGA32U4	16MHz	28KB	2.50KB
metro	Adafruit Metro	No	AT-MEGA328P	16MHz	31.50KB	2KB
protrinket3	Adafruit Pro Trinket 3V/12MHz (USB)	No	AT-MEGA328P	12MHz	28KB	2KB
protrinket3ftdi	Adafruit Pro Trinket 3V/12MHz (FTDI)	No	AT-MEGA328P	12MHz	28KB	2KB
protrinket5	Adafruit Pro Trinket 5V/16MHz (USB)	No	AT-MEGA328P	16MHz	28KB	2KB
protrinket5ftdi	Adafruit Pro Trinket 5V/16MHz (FTDI)	No	AT-MEGA328P	16MHz	28KB	2KB
trinket3	Adafruit Trinket 3V/8MHz	No	ATTINY85	8MHz	8KB	512B
trinket5	Adafruit Trinket 5V/16MHz	No	ATTINY85	16MHz	8KB	512B

Alorium Technology

ID	Name	Debug	MCU	Frequency	Flash	RAM
alorium_xlr8	Alorium XLR8	No	ATMEGA328P	16MHz	31.50KB	2KB

Anarduino

ID	Name	Debug	MCU	Frequency	Flash	RAM
miniwireless	Anarduino MiniWireless	No	ATMEGA328P	16MHz	31.50KB	2KB

Arduboy

ID	Name	Debug	MCU	Frequency	Flash	RAM
arduboy	Arduboy	No	ATMEGA32U4	16MHz	28KB	2.50KB
arduboy_devkit	Arduboy DevKit	No	ATMEGA32U4	16MHz	28KB	2.50KB

Arduino

ID	Name	Debug	MCU	Frequency
LilyPadUSB	Arduino LilyPad USB	No	ATMEGA32U4	8MHz
atmega328pb	Atmel ATmega328PB	No	ATMEGA328PB	16MHz
atmegangatmega168	Arduino NG or older ATmega168	No	ATMEGA168	16MHz
atmegangatmega8	Arduino NG or older ATmega8	No	ATMEGA8	16MHz
btatmega168	Arduino BT ATmega168	No	ATMEGA168	16MHz
btatmega328	Arduino BT ATmega328	No	ATMEGA328P	16MHz
chiwawa	Arduino Industrial 101	No	ATMEGA32U4	16MHz
diecimilaatmega168	Arduino Duemilanove or Diecimila ATmega168	No	ATMEGA168	16MHz
diecimilaatmega328	Arduino Duemilanove or Diecimila ATmega328	No	ATMEGA328P	16MHz
esplora	Arduino Esplora	No	ATMEGA32U4	16MHz
ethernet	Arduino Ethernet	No	ATMEGA328P	16MHz
fio	Arduino Fio	No	ATMEGA328P	8MHz
leonardo	Arduino Leonardo	No	ATMEGA32U4	16MHz
leonardoeth	Arduino Leonardo ETH	No	ATMEGA32U4	16MHz
lilypadatmega168	Arduino LilyPad ATmega168	No	ATMEGA168	8MHz
lilypadatmega328	Arduino LilyPad ATmega328	No	ATMEGA328P	8MHz
megaADK	Arduino Mega ADK	No	ATMEGA2560	16MHz
megaatmega1280	Arduino Mega or Mega 2560 ATmega1280	No	ATMEGA1280	16MHz
megaatmega2560	Arduino Mega or Mega 2560 ATmega2560 (Mega 2560)	No	ATMEGA2560	16MHz
micro	Arduino Micro	No	ATMEGA32U4	16MHz
miniatmega168	Arduino Mini ATmega168	No	ATMEGA168	16MHz
miniatmega328	Arduino Mini ATmega328	No	ATMEGA328P	16MHz
nanoatmega168	Arduino Nano ATmega168	No	ATMEGA168	16MHz
nanoatmega328	Arduino Nano ATmega328	No	ATMEGA328P	16MHz
pro16MHzatmega168	Arduino Pro or Pro Mini ATmega168 (5V, 16 MHz)	No	ATMEGA168	16MHz
pro16MHzatmega328	Arduino Pro or Pro Mini ATmega328 (5V, 16 MHz)	No	ATMEGA328P	16MHz
pro8MHzatmega168	Arduino Pro or Pro Mini ATmega168 (3.3V, 8 MHz)	No	ATMEGA168	8MHz
pro8MHzatmega328	Arduino Pro or Pro Mini ATmega328 (3.3V, 8 MHz)	No	ATMEGA328P	8MHz
robotControl	Arduino Robot Control	No	ATMEGA32U4	16MHz
robotMotor	Arduino Robot Motor	No	ATMEGA32U4	16MHz
uno	Arduino Uno	No	ATMEGA328P	16MHz
yun	Arduino Yun	No	ATMEGA32U4	16MHz
yunmini	Arduino Yun Mini	No	ATMEGA32U4	16MHz

Atmel

ID	Name	Debug	MCU	Frequency	Flash	RAM
attiny13	Generic ATTiny13	No	ATTINY13	9MHz	1KB	64B
attiny1634	Generic ATTiny1634	No	ATTINY1634	8MHz	16KB	1KB
attiny167	Generic ATTiny167	No	ATTINY167	8MHz	16KB	512B
attiny2313	Generic ATTiny2313	No	ATTINY2313	8MHz	2KB	128B
attiny24	Generic ATTiny24	No	ATTINY24	8MHz	2KB	128B
attiny25	Generic ATTiny25	No	ATTINY25	8MHz	2KB	128B
attiny261	Generic ATTiny261	No	ATTINY261	8MHz	2KB	128B
attiny4313	Generic ATTiny4313	No	ATTINY4313	8MHz	4KB	256B
attiny44	Generic ATTiny44	No	ATTINY44	8MHz	4KB	256B
attiny441	Generic ATTiny441	No	ATTINY441	8MHz	4KB	256B
attiny45	Generic ATTiny45	No	ATTINY45	8MHz	4KB	256B
attiny461	Generic ATTiny461	No	ATTINY461	8MHz	4KB	256B
attiny48	Generic ATTiny48	No	ATTINY48	8MHz	4KB	256B
attiny828	Generic ATTiny828	No	ATTINY828	8MHz	8KB	512B
attiny84	Generic ATTiny84	No	ATTINY84	8MHz	8KB	512B
attiny841	Generic ATTiny841	No	ATTINY841	8MHz	8KB	512B
attiny85	Generic ATTiny85	No	ATTINY85	8MHz	8KB	512B
attiny861	Generic ATTiny861	No	ATTINY861	8MHz	8KB	512B
attiny87	Generic ATTiny87	No	ATTINY87	8MHz	8KB	512B
attiny88	Generic ATTiny88	No	ATTINY88	8MHz	8KB	512B
usbasp	USBasp stick	No	ATMEGA8	12MHz	8KB	1KB

BQ

ID	Name	Debug	MCU	Frequency	Flash	RAM
zumbt328	BQ ZUM BT-328	No	ATMEGA328P	16MHz	28KB	2KB

BitWizard

ID	Name	Debug	MCU	Frequency	Flash	RAM
raspduino	BitWizard Raspduino	No	ATMEGA328P	16MHz	30KB	2KB

Controllino

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
controllino_maxi	Controllino Maxi	No	AT-MEGA2560	16MHz	248KB	8KB
controllino_maxi_automation	Controllino Maxi Automation	No	AT-MEGA2560	16MHz	248KB	8KB
controllino_mega	Controllino Mega	No	AT-MEGA2560	16MHz	248KB	8KB
controllino_mini	Controllino Mini	No	AT-MEGA328P	16MHz	31.50KB	2KB

Digistump

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
digispark-pro	Digispark Pro	No	AT-TINY167	16MHz	14.50KB	512B
digispark-pro32	Digispark Pro (32 byte buffer)	No	AT-TINY167	16MHz	14.50KB	512B
digispark-pro64	Digispark Pro (16 MHz) (64 byte buffer)	No	AT-TINY167	16MHz	14.50KB	512B
digispark-tiny	Digispark USB	No	AT-TINY85	16MHz	5.87KB	512B

Dwengo

ID	Name	Debug	MCU	Frequency	Flash	RAM
dwenguino	Dwenguino	No	AT90USB646	16MHz	60KB	2KB

Elektor

ID	Name	Debug	MCU	Frequency	Flash	RAM
elektor_uno_r4	Elektor Uno R4	No	ATMEGA328PB	16MHz	31.50KB	2KB

Engduino

ID	Name	Debug	MCU	Frequency	Flash	RAM
engduinov3	Engduino 3	No	ATMEGA32U4	8MHz	28KB	2.50KB

EnviroDIY

ID	Name	Debug	MCU	Frequency	Flash	RAM
mayfly	EnviroDIY Mayfly	No	ATMEGA1284P	8MHz	127KB	16KB

LightUp

ID	Name	Debug	MCU	Frequency	Flash	RAM
lightup	LightUp	No	ATMEGA32U4	8MHz	28KB	2.50KB

Linino

ID	Name	Debug	MCU	Frequency	Flash	RAM
one	Linino One	No	ATMEGA32U4	16MHz	28KB	2.50KB

LowPowerLab

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
mightyhat	LowPowerLab MightyHat	No	ATMEGA328P	16MHz	31KB	2KB
moteino	LowPowerLab Moteino	No	ATMEGA328P	16MHz	31.50KB	2KB
moteinomega	LowPowerLab MoteinoMEGA	No	AT- MEGA1284P	16MHz	127KB	16KB

McuDude

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
mightycore1284	MightyCore AT- mega1284	No	AT- MEGA1284P	16MHz	127KB	16KB
mightycore16	MightyCore ATmega16	No	ATMEGA16	16MHz	15.50KB	1KB
mightycore164	MightyCore ATmega164	No	ATMEGA164P	16MHz	15.50KB	1KB
mightycore32	MightyCore ATmega32	No	ATMEGA32	16MHz	31.50KB	2KB
mightycore324	MightyCore ATmega324	No	ATMEGA324P	16MHz	31.50KB	2KB
mightycore644	MightyCore ATmega644	No	ATMEGA644P	16MHz	63KB	4KB
mightycore8535	MightyCore AT- mega8535	No	ATMEGA16	16MHz	7.50KB	512B

MediaTek Labs

ID	Name	Debug	MCU	Frequency	Flash	RAM
smart7688	LinkIt Smart 7688 Duo	No	ATMEGA32U4	8MHz	28KB	2.50KB

Microchip

ID	Name	Debug	MCU	Frequency	Flash	RAM
at90pwm216	Atmel AT90PWM216	No	AT90PWM216	16MHz	16KB	1KB
at90pwm316	Atmel AT90PWM316	No	AT90PWM316	16MHz	16KB	1KB

Micoduino

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
1284p16m	Micoduino Core+ (ATmega1284P@16M,5V)	No	AT-MEGA1284P	16MHz	127KB	16KB
1284p8m	Micoduino Core+ (ATmega1284P@8M,3.3V)	No	AT-MEGA1284P	8MHz	127KB	16KB
168pa16m	Micoduino Core (Atmega168PA@16M,5V)	No	AT-MEGA168P	16MHz	15.50KB	1KB
168pa8m	Micoduino Core (Atmega168PA@8M,3.3V)	No	AT-MEGA168P	8MHz	15.50KB	1KB
328p16m	Micoduino Core (Atmega328P@16M,5V)	No	AT-MEGA328P	16MHz	31.50KB	2KB
328p8m	Micoduino Core (Atmega328P@8M,3.3V)	No	AT-MEGA328P	8MHz	31.50KB	2KB
32u416m	Micoduino Core USB (ATmega32U4@16M,5V)	No	AT-MEGA32U4	16MHz	28KB	2.50KB
644pa16m	Micoduino Core+ (Atmega644PA@16M,5V)	No	AT-MEGA644P	16MHz	63KB	4KB
644pa8m	Micoduino Core+ (Atmega644PA@8M,3.3V)	No	AT-MEGA644P	8MHz	63KB	4KB

OpenEnergyMonitor

ID	Name	Debug	MCU	Frequency	Flash	RAM
emonpi	OpenEnergyMonitor emonPi	No	ATMEGA328P	16MHz	30KB	2KB

PanStamp

ID	Name	Debug	MCU	Frequency	Flash	RAM
panStampAVR	PanStamp AVR	No	ATMEGA328P	8MHz	31.50KB	2KB

Pinoccio

ID	Name	Debug	MCU	Frequency	Flash	RAM
pinoccio	Pinoccio Scout	No	ATMEGA256RFR2	16MHz	248KB	32KB

Pololu Corporation

ID	Name	Debug	MCU	Frequency	Flash	RAM
a-star32U4	Pololu A-Star 32U4	No	ATMEGA32U4	16MHz	28KB	2.50KB

Punch Through

ID	Name	Debug	MCU	Frequency	Flash	RAM
lightblue-bean	LightBlue Bean	No	ATMEGA328P	8MHz	31.50KB	2KB
lightblue-beanplus	LightBlue Bean+	No	ATMEGA328P	16MHz	31.50KB	2KB

Quirkbot

ID	Name	Debug	MCU	Frequency	Flash	RAM
quirkbot	Quirkbot	No	ATMEGA32U4	8MHz	28KB	2.50KB

RedBearLab

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
blend	RedBearLab Blend	No	AT-MEGA32U4	16MHz	28KB	2.50KB
blendmicro16	RedBearLab Blend Micro 3.3V/16MHz (overclock)	No	AT-MEGA32U4	16MHz	28KB	2.50KB
blendmicro8	RedBearLab Blend Micro 3.3V/8MHz	No	AT-MEGA32U4	8MHz	28KB	2.50KB

RepRap

ID	Name	Debug	MCU	Frequency	Flash	RAM
reprap_rambo	RepRap RAMBo	No	ATMEGA2560	16MHz	252KB	8KB

SODAQ

ID	Name	Debug	MCU	Frequency	Flash	RAM
sodaq_galora	SODAQ GaLoRa	No	ATMEGA1284P	8MHz	127KB	16KB
sodaq_mbili	SODAQ Mbili	No	ATMEGA1284P	8MHz	127KB	16KB
sodaq_moja	SODAQ Moja	No	ATMEGA328P	8MHz	31.50KB	2KB
sodaq_ndogo	SODAQ Ndogo	No	ATMEGA1284P	8MHz	127KB	16KB
sodaq_tatu	SODAQ Tatu	No	ATMEGA1284P	8MHz	127KB	16KB

Sanguino

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
sanguino_atmega1280	Sanguino ATmega1284P (8MHz)	No	AT-MEGA1284P	8MHz	127KB	16KB
sanguino_atmega1280	Sanguino ATmega1284P (16MHz)	No	AT-MEGA1284P	16MHz	127KB	16KB
sanguino_atmega644	Sanguino ATmega644 or ATmega644A (16 MHz)	No	AT-MEGA644	16MHz	63KB	4KB
sanguino_atmega644	Sanguino ATmega644 or ATmega644A (8 MHz)	No	AT-MEGA644	8MHz	63KB	4KB
sanguino_atmega644	Sanguino ATmega644P or ATmega644PA (16 MHz)	No	AT-MEGA644P	16MHz	63KB	4KB
sanguino_atmega644	Sanguino ATmega644P or ATmega644PA (8 MHz)	No	AT-MEGA644P	8MHz	63KB	4KB

SeeedStudio

ID	Name	Debug	MCU	Frequency	Flash	RAM
seeeduino	Seeeduino	No	ATMEGA328P	16MHz	31.50KB	2KB

SparkFun

ID	Name	Debug	MCU	Frequency	Flash	RAM
sparkfun_digital sandbox	SparkFun Digital Sandbox	No	AT-MEGA328P	8MHz	31.50KB	2KB
sparkfun_fio v3	SparkFun Fio V3 3.3V/8MHz	No	AT-MEGA32U4	8MHz	28KB	2.50KB
sparkfun_makey makey	SparkFun Makey Makey	No	AT-MEGA32U4	16MHz	28KB	2.50KB
sparkfun_megamini	SparkFun Mega Pro Mini 3.3V	No	AT-MEGA2560	8MHz	252KB	8KB
sparkfun_megapro16m	SparkFun Mega Pro 5V/16MHz	No	AT-MEGA2560	16MHz	248KB	8KB
sparkfun_megapro8m	SparkFun Mega Pro 3.3V/8MHz	No	AT-MEGA2560	8MHz	252KB	8KB
sparkfun_promicro16	SparkFun Pro Micro 5V/16MHz	No	AT-MEGA32U4	16MHz	28KB	2.50KB
sparkfun_promicro8	SparkFun Pro Micro 3.3V/8MHz	No	AT-MEGA32U4	8MHz	28KB	2.50KB
sparkfun_qduinomin	SparkFun Qduino Mini	No	AT-MEGA32U4	8MHz	28KB	2.50KB
sparkfun_redboard	SparkFun RedBoard	No	AT-MEGA328P	16MHz	31.50KB	2KB
sparkfun_satmega128rfa1	SparkFun ATmega128RFA1 Dev Board	No	AT-MEGA128RFA1	16MHz	16KB	124KB
sparkfun_serial7seg	SparkFun Serial 7-Segment Display	No	AT-MEGA328P	8MHz	31.50KB	2KB
uvview	SparkFun MicroView	No	AT-MEGA328P	16MHz	31.50KB	2KB

SpellFoundry

ID	Name	Debug	MCU	Frequency	Flash	RAM
sleepypi	SpellFoundry Sleepy Pi 2	No	ATMEGA328P	8MHz	30KB	2KB

The Things Network

ID	Name	Debug	MCU	Frequency	Flash	RAM
the_things_uno	The Things Uno	No	ATMEGA32U4	16MHz	28KB	2.50KB

TinyCircuits

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
tinyduino	TinyCircuits TinyDuino Processor Board	No	AT-MEGA328P	8MHz	30KB	2KB
tinylily	TinyCircuits TinyLily Mini Processor	No	AT-MEGA328P	8MHz	30KB	2KB

Wicked Device

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
wildfirev2	Wicked Device WildFire V2	No	AT-MEGA1284P	16MHz	120.00KB	16KB
wildfirev3	Wicked Device WildFire V3	No	AT-MEGA1284P	16MHz	127KB	16KB

makerlab.mx

ID	Name	Debug	MCU	Frequency	Flash	RAM
altair	Altair	No	ATMEGA256RFR2	16MHz	248KB	32KB

nicai-systems

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
bob3	nicai-systems BOB3 coding bot	No	ATMEGA88	8MHz	8KB	1KB
nibo2	nicai-systems NIBO 2 robot	No	AT-MEGA128	16MHz	128KB	4KB
nibobee	nicai-systems NIBObee robot	No	ATMEGA16	15MHz	16KB	1KB
nibobee_1284	nicai-systems NIBObee robot with Tuning Kit	No	AT-MEGA1284P	20MHz	128KB	16KB
niboburger	nicai-systems NIBO burger robot	No	ATMEGA16	15MHz	16KB	1KB
niboburger_1284	nicai-systems NIBO burger robot with Tuning Kit	No	AT-MEGA1284P	20MHz	128KB	16KB

ubIQio

ID	Name	Debug	MCU	Frequency	Flash	RAM
ardhat	ubIQio Ardhat	No	ATMEGA328P	16MHz	31.50KB	2KB

Atmel SAM

`platform = atmelsam`

Atmel | SMART offers Flash- based ARM products based on the ARM Cortex-M0+, Cortex-M3 and Cortex-M4 architectures, ranging from 8KB to 2MB of Flash including a rich peripheral and feature mix.

For more detailed information please visit [vendor site](#).

Contents

- [*Examples*](#)
- [*Debugging*](#)
- [*Stable and upstream versions*](#)
- [*Packages*](#)
- [*Frameworks*](#)
- [*Boards*](#)

Examples

Examples are listed from Atmel SAM development platform repository:

- [arduino-blink](#)
- [arduino-external-libs](#)
- [arduino-internal-libs](#)
- [arduino-web-thing-led](#)
- [mbed-blink](#)
- [mbed-dsp](#)
- [mbed-events](#)
- [mbed-serial](#)
- [simba-blink](#)

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- [*Debug Tools*](#)
 - [*On-Board Debug Tools*](#)
 - [*External Debug Tools*](#)

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Banks listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
mzeropro	Arduino M0 Pro (Programming/Debug Port)	<i>CMSIS-DAP</i> (on-board), <i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAMD21G18MHz	256KB	32KB	
samd21_xpro	Atmel SAMD21-XPRO	<i>CMSIS-DAP</i> (on-board), <i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAMD21J18MHz	256KB	32KB	
samd21g18a	Atmel ATSAMW25-XPRO	<i>CMSIS-DAP</i> (on-board), <i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAMD21G18MHz	256KB	32KB	
saml21_xpro	Atmel SAML21-XPRO-B	<i>CMSIS-DAP</i> (on-board), <i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAML21J18MHz	256KB	32KB	
samr21_xpro	Atmel ATSAMR21-XPRO	<i>CMSIS-DAP</i> (on-board), <i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAMR21G18MHz	256KB	32KB	
zero	Arduino Zero (Programming/Debug Port)	<i>CMSIS-DAP</i> (on-board), <i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAMD21G18MHz	256KB	32KB	

External Debug Tools

Banks listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU
adafruit_circuitplayground_m0	Adafruit Circuit Playground Express	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
adafruit_feather_m0	Adafruit Feather M0	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
adafruit_feather_m0_express	Adafruit Feather M0 Express	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
adafruit_feather_m4	Adafruit Feather M4 (SAMD51)	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
adafruit_gemma_m0	Adafruit Gemma M0	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
adafruit_itsybitsy_m0	Adafruit ItsyBitsy M0	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
adafruit_itsybitsy_m4	Adafruit ItsyBitsy M4 (SAMD51)	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
adafruit_metro_m0	Adafruit Metro M0 Express	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
adafruit_metro_m4	Adafruit Metro M4 (SAMD51)	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
adafruit_pirkey	Adafruit pIRkey	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
adafruit_trinket_m0	Adafruit Trinket M0	<i>Atmel-ICE</i> , <i>Black Magic Probe</i> , <i>J-LINK</i>	SAM
digix	Digistump DigiX	<i>Black Magic Probe</i> , <i>J-LINK</i>	AT9
due	Arduino Due (Programming Port)	<i>Black Magic Probe</i> , <i>J-LINK</i>	AT9
dueUSB	Arduino Due (USB Native Port)	<i>Black Magic Probe</i> , <i>J-LINK</i>	AT9

Table 2 – continued from previous page

ID	Name	Debug	MCP
macchina2	Macchina M2	<i>Black Magic Probe, J-LINK</i>	AT91SAM9263
mkr1000USB	Arduino MKR1000	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
mkrfox1200	Arduino MKR FOX 1200	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
mkrgsm1400	Arduino MKR GSM 1400	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
mkewan1300	Arduino MKR WAN 1300	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
mkrzero	Arduino MKRZERO	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
mzeroUSB	Arduino M0	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
mzeroproub	Arduino M0 Pro (Native USB Port)	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
sainSmartDue	SainSmart Due (Programming Port)	<i>Black Magic Probe, J-LINK</i>	AT91SAM9263
sainSmartDueUSB	SainSmart Due (USB Native Port)	<i>Black Magic Probe, J-LINK</i>	AT91SAM9263
sodaq_autonomo	SODAQ Autonomo	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
sodaq_explorer	SODAQ ExpLoRer	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
sodaq_one	SODAQ ONE	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
sparkfun_samd21_dev_usb	SparkFun SAMD21 Dev Breakout	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
sparkfun_samd21_mini_usb	SparkFun SAMD21 Mini Breakout	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
tian	Arduino Tian	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E
zeroUSB	Arduino Zero (USB Native Port)	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAM3X8E

Stable and upstream versions

You can switch between stable releases of Atmel SAM development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = atmelsam
board = ...

; Custom stable version
[env:custom_stable]
platform = atmelsam@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-atmelsam.git
board = ...
```

Packages

Name	Description
framework-arduinosam	Arduino Wiring-based Framework (SAM Core, 1.6)
framework-mbed	mbed Framework
framework-simba	Simba Framework
tool-avrdude	AVRDUDE
tool-bossac	BOSSA CLI
tool-openocd	OpenOCD
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Arduin	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.
Simba	Simba is an RTOS and build framework. It aims to make embedded programming easy and portable.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or [PlatformIO Boards Explorer](#)
- For more detailed board information please scroll tables below by horizontal.

Adafruit

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
adafruit_circuitplayground_express	Adafruit Circuit Playground Express	Yes	SAMD21G18A48MHz	256KB	32KB	
adafruit_feather_m0	Adafruit Feather M0	Yes	SAMD21G18A48MHz	256KB	32KB	
adafruit_feather_m0_express	Adafruit Feather M0 Express	Yes	SAMD21G18A48MHz	256KB	32KB	
adafruit_feather_m4	Adafruit Feather M4 (SAMD51)	Yes	SAMD51J19A120MHz	496KB	192KB	
adafruit_gemma_m0	Adafruit Gemma M0	Yes	SAMD21E18A48MHz	256KB	32KB	
adafruit_itsybitsy_m0	Adafruit ItsyBitsy M0	Yes	SAMD21G18A48MHz	256KB	32KB	
adafruit_itsybitsy_m4	Adafruit ItsyBitsy M4 (SAMD51)	Yes	SAMD51J19A120MHz	496KB	192KB	
adafruit_metro_m0	Adafruit Metro M0 Express	Yes	SAMD21G18A48MHz	256KB	32KB	
adafruit_metro_m4	Adafruit Metro M4 (SAMD51)	Yes	SAMD51J19A120MHz	496KB	192KB	
adafruit_pirkey	Adafruit pIRkey	Yes	SAMD21E18A48MHz	256KB	32KB	
adafruit_trinket_m0	Adafruit Trinket M0	Yes	SAMD21E18A48MHz	256KB	32KB	

Arduino

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
due	Arduino Due (Programming Port)	Yes	AT91SAM3X8E	84MHz	512KB	32KB
dueUSB	Arduino Due (USB Native Port)	Yes	AT91SAM3X8E	84MHz	512KB	32KB
mkr1000USB	Arduino MKR1000	Yes	SAMD21G18A	48MHz	256KB	32KB
mkrfox1200	Arduino MKR FOX 1200	Yes	SAMD21G18A	48MHz	256KB	32KB
mkrgsm1400	Arduino MKR GSM 1400	Yes	SAMD21G18A	48MHz	256KB	32KB
mkewan1300	Arduino MKR WAN 1300	Yes	SAMD21G18A	48MHz	256KB	32KB
mkrzero	Arduino MKRZERO	Yes	SAMD21G18A	48MHz	256KB	32KB
mzeroUSB	Arduino M0	Yes	SAMD21G18A	48MHz	256KB	32KB
mzeropro	Arduino M0 Pro (Programming/Debug Port)	Yes	SAMD21G18A	48MHz	256KB	32KB
mzeroprouse	Arduino M0 Pro (Native USB Port)	Yes	SAMD21G18A	48MHz	256KB	32KB
tian	Arduino Tian	Yes	SAMD21G18A	48MHz	256KB	32KB
zero	Arduino Zero (Programming/Debug Port)	Yes	SAMD21G18A	48MHz	256KB	32KB
zeroUSB	Arduino Zero (USB Native Port)	Yes	SAMD21G18A	48MHz	256KB	32KB

Atmel

ID	Name	Debug	MCU	Frequency	Flash	RAM
samd21_xpro	Atmel SAMD21-XPRO	Yes	SAMD21J18A	48MHz	256KB	32KB
samd21g18a	Atmel ATSAMW25-XPRO	Yes	SAMD21G18A	48MHz	256KB	32KB
saml21_xpro_b	Atmel SAML21-XPRO-B	Yes	SAML21J18B	48MHz	256KB	32KB
samr21_xpro	Atmel ATSAMR21-XPRO	Yes	SAMR21G18A	48MHz	256KB	32KB

Digistump

ID	Name	Debug	MCU	Frequency	Flash	RAM
digix	Digistump DigiX	Yes	AT91SAM3X8E	84MHz	512KB	28KB

Macchina

ID	Name	Debug	MCU	Frequency	Flash	RAM
macchina2	Macchina M2	Yes	AT91SAM3X8E	84MHz	512KB	32KB

SODAQ

ID	Name	Debug	MCU	Frequency	Flash	RAM
sodaq_autonomo	SODAQ Autonomo	Yes	SAMD21J18A	48MHz	256KB	32KB
sodaq_explorer	SODAQ ExpLoRer	Yes	SAMD21J18A	48MHz	256KB	32KB
sodaq_one	SODAQ ONE	Yes	SAMD21G18A	48MHz	256KB	32KB

SainSmart

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
sainSmartDue	SainSmart Due (Programming Port)	Yes	AT91SAM3X8E	84MHz	512KB	32KB
sainSmartDueUSB	SainSmart Due (USB Native Port)	Yes	AT91SAM3X8E	84MHz	512KB	32KB

SparkFun

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
sparkfun_samd21_dev_usb	SparkFun SAMD21 Dev Breakout	Yes	SAMD21G18A	48MHz	256KB	32KB
sparkfun_samd21_mini	SparkFun SAMD21 Mini Breakout	Yes	SAMD21G18A	48MHz	256KB	32KB

Espressif 32

`platform = espressif32`

Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

For more detailed information please visit [vendor site](#).

Contents

- [*Configuration*](#)
- [*Examples*](#)
- [*Debugging*](#)
- [*Stable and upstream versions*](#)
- [*Packages*](#)
- [*Frameworks*](#)
- [*Boards*](#)

Configuration

- [*CPU Frequency*](#)
- [*FLASH Frequency*](#)
- [*FLASH Mode*](#)
- [*External RAM \(PSRAM\)*](#)
- [*Debug Level*](#)
- [*Upload Speed*](#)
- [*Enable C++ exceptions*](#)
- [*Partition Tables*](#)
- [*Embedding Binary Data*](#)
- [*Uploading files to file system SPIFFS*](#)
- [*Over-the-Air \(OTA\) update*](#)
 - [*Using JFrog Bintray \(free and secure Cloud solution\)*](#)
 - [*Using built-in Local solution*](#)
 - * [*Authentication and upload options*](#)
- [*Using Arduino Framework with Staging version*](#)
- [*Arduino Core Wiki*](#)

CPU Frequency

See `board_build.f_cpu` option from *Project Configuration File platformio.ini*

```
[env:myenv]
; set frequency to 160MHz
board_build.f_cpu = 160000000L
```

FLASH Frequency

Please use `board_build.f_flash` option from *Project Configuration File platformio.ini* to change a value. Possible values:

- 40000000L (default)
- 80000000L

```
[env:myenv]
; set frequency to 80MHz
board_build.f_flash = 80000000L
```

FLASH Mode

Flash chip interface mode. This parameter is stored in the binary image header, along with the flash size and flash frequency. The ROM bootloader in the ESP chip uses the value of these parameters in order to know how to talk to the flash chip.

Please use `board_build.flash_mode` option from *Project Configuration File platformio.ini* to change a value. Possible values:

- qio
- qout
- dio
- dout

```
[env:myenv]
board_build.flash_mode = qio
```

External RAM (PSRAM)

You can enable external RAM using the next extra `build_flags` in *Project Configuration File platformio.ini* depending on a framework type.

Framework *Arduino*:

```
[env:myenv]
platform = espressif32
framework = arduino
board = ...
build_flags =
    -DBOARD_HAS_PSRAM
    -mfix-esp32-psram-cache-issue
```

Framework *ESP-IDF*:

```
[env:myenv]
platform = espressif32
framework = espidf
board = ...
build_flags =
    -DCONFIG_SPIRAM_CACHE_WORKAROUND
```

More details are located in the official ESP-IDF documentation - [Support for external RAM](#).

Debug Level

Please use one of the next *build_flags* to change debug level. A *build_flags* option could be used only the one time per build environment. If you need to specify more flags, please separate them with a new line or space.

Actual information is available in [Arduino for ESP32 Board Manifest](#). Please scroll to `esp32.menu.DebugLevel` section.

```
[env:myenv]
platform = ...
board = ...
framework = arduino

;;;; Possible options ;;;;

; None
build_flags = -DCORE_DEBUG_LEVEL=0

; Error
build_flags = -DCORE_DEBUG_LEVEL=1

; Warn
build_flags = -DCORE_DEBUG_LEVEL=2

; Info
build_flags = -DCORE_DEBUG_LEVEL=3

; Debug
build_flags = -DCORE_DEBUG_LEVEL=4

; Verbose
build_flags = -DCORE_DEBUG_LEVEL=5
```

Upload Speed

You can set custom upload speed using *upload_speed* option from *Project Configuration File platformio.ini*

```
[env:myenv]
upload_speed = 9600
```

Enable C++ exceptions

Please add `-D PIO_FRAMEWORK_ESP_IDF_ENABLE_EXCEPTIONS` to *build_flags* of *Project Configuration File platformio.ini* to enable C++ exceptions for [ESP-IDF](#).

See project example.

Partition Tables

You can create a custom partitions table (CSV) following [ESP32 Partition Tables](#) documentation. PlatformIO uses **default partition tables** depending on a *framework* type:

- “`default.csv`” for Arduino
- “`partitions_singleapp.csv`” for ESP-IDF

To override default table please use `board_build.partitions` option in *Project Configuration File platformio.ini*.

Warning: SPIFFS partition **MUST** have configured “Type” as “data” and “SubType” as “spiffs”. For example,
`spiffs, data, spiffs, 0x291000, 1M,`

Examples:

```
; 1) A "partitions_custom.csv" in the root of project directory
[env:custom_table]
board_build.partitions = partitions_custom.csv

; 2) Switch between built-in tables
; https://github.com/espressif/arduino-esp32/tree/master/tools/partitions
; https://github.com/espressif/esp-idf/tree/master/components/partition_table
[env:custom_builtin_table]
board_build.partitions = no_ota.csv
```

Embedding Binary Data

Sometimes you have a file with some binary or text data that you'd like to make available to your program - but you don't want to reformat the file as C source.

You can set a macro (define) `COMPONENT_EMBED_FILES` using *build_flags* in *Project Configuration File platformio.ini*, giving the names of the files to embed in this way:

```
[env:myenv]
platform = espressif32
board = ...
build_flags =
    -DCOMPONENT_EMBED_TXTFILES=src/private.pem.key:src/certificate.pem.crt:src/aws-root-ca.pem
```

Multiple files are allowed and should be split by colon - `:`.

The file's contents will be added to the `.rodata` section in flash, and are available via symbol names as follows:

```
extern const uint8_t aws_root_ca_pem_start[] asm("_binary_src_aws_root_ca_pem_start");
extern const uint8_t aws_root_ca_pem_end[] asm("_binary_src_aws_root_ca_pem_end");
extern const uint8_t certificate_pem_crt_start[]_
→asm("_binary_src_certificate_pem_crt_start");
extern const uint8_t certificate_pem_crt_end[]_
→asm("_binary_src_certificate_pem_crt_end");
extern const uint8_t private_pem_key_start[] asm("_binary_src_private_pem_key_start");
extern const uint8_t private_pem_key_end[] asm("_binary_src_private_pem_key_end");
```

The names are generated from the full name of the file, as given in `COMPONENT_EMBED_FILES`. Characters /, ., etc. are replaced with underscores. The `_binary + _nested_folder` prefix in the symbol name is added by “objcopy” and is the same for both text and binary files.

See full example with embedding Amazon AWS certificates:

- <https://github.com/platformio/platform-espressif32/tree/develop/examples/espidf-aws-iot>

Uploading files to file system SPIFFS

1. Initialize project `platformio init` (if you have not initialized yet)
2. Create `data` folder (it should be on the same level as `src` folder) and put files here. Also, you can specify own location for `data_dir`
3. Run `buildfs` or `uploadfs` target using `platformio run --target` command.

To upload SPIFFS image using OTA update please specify `upload_port` / `--upload-port` as IP address or mDNS host name (ending with the `*.local`).

Examples:

- SPIFFS for Arduino
- SPIFFS for ESP-IDF

Over-the-Air (OTA) update

Using JFrog Bintray (free and secure Cloud solution)

- Video and presentation - swampUP: Over-The-Air (OTA) firmware upgrades for Internet of Things devices with PlatformIO and JFrog Bintray
- Demo source code: <https://github.com/platformio/bintray-secure-ota>

Using built-in Local solution

Demo code for:

- Arduino
- ESP-IDF

There are 2 options how to upload firmware OTA:

- Directly specify `platformio run --upload-port` in command line

```
platformio run --target upload --upload-port IP_ADDRESS_HERE or mDNS_NAME.local
```

- Specify `upload_port` option in *Project Configuration File platformio.ini*

```
[env:myenv]
upload_port = IP_ADDRESS_HERE or mDNS_NAME.local
```

For example,

- `platformio run -t upload --upload-port 192.168.0.255`
- `platformio run -t upload --upload-port myesp32.local`

Authentication and upload options

You can pass additional options/flags to OTA uploader using `upload_flags` option in *Project Configuration File platformio.ini*

```
[env:myenv]
upload_flags = --port=3232
```

Available flags

- `--port=ESP_PORT` ESP32 OTA Port. Default 3232
- `--auth=AUTH` Set authentication password
- `--spiffs` Use this option to transmit a SPIFFS image and do not flash the module

For the full list with available options please run

```
~/platformio/packages/tool-espotapy/espotapy.py -h

Usage: spotapy [options]

Transmit image over the air to the esp32 module with OTA support.

Options:
  -h, --help           show this help message and exit

Destination:
  -i ESP_IP, --ip=ESP_IP
                      ESP32 IP Address.
  -p ESP_PORT, --port=ESP_PORT
                      ESP32 ota Port. Default 3232

Authentication:
  -a AUTH, --auth=AUTH
                      Set authentication password.

Image:
  -f FILE, --file=FILE
                      Image file.
  -s, --spiffs        Use this option to transmit a SPIFFS image and do not
                      flash the module.

Output:
```

(continues on next page)

(continued from previous page)

<code>-d, --debug</code>	Show debug output. And override loglevel with debug.
<code>-r, --progress</code>	Show progress output. Does not work for ArduinoIDE

Using Arduino Framework with Staging version

PlatformIO will install the latest Arduino Core for ESP32 from <https://github.com/espressif/arduino-esp32>. The [Git](#) should be installed in a system. To update Arduino Core to the latest revision, please open [PlatformIO IDE](#) and navigate to PIO Home > Platforms > Updates.

1. Please install [PlatformIO IDE](#)
2. Initialize a new project, open *Project Configuration File platformio.ini* and set *platform* to `https://github.com/platformio/platform-espressif32.git#feature/stage`. For example,

```
[env:esp32dev]
platform = https://github.com/platformio/platform-espressif32.git#feature/stage
board = esp32dev
framework = arduino
```

3. Try to build project
4. If you see build errors, then try to build this project using the same `stage` with Arduino IDE
5. If it works with Arduino IDE but doesn't work with PlatformIO, then please [file](#) new issue with attached information:
 - test project/files
 - detailed log of build process from Arduino IDE (please copy it from console to <https://hastebin.com>)
 - detailed log of build process from PlatformIO Build System (please copy it from console to <https://hastebin.com>)

Arduino Core Wiki

Tips, tricks and common problems: <http://desire.giesecke.tk/index.php/2018/01/30/esp32-wiki-entries/>

Examples

Examples are listed from Espressif 32 development platform repository:

- `arduino-blink`
- `arduino-wifiscan`
- `espidf-aws-iot`
- `espidf-ble-adv`
- `espidf-coap-server`
- `espidf-exceptions`
- `espidf-hello-world`
- `espidf-http-request`
- `espidf-peripherals-uart`

- espidf-storage-sdcard
 - pumbaa-blink
 - simba-blink

Debugging

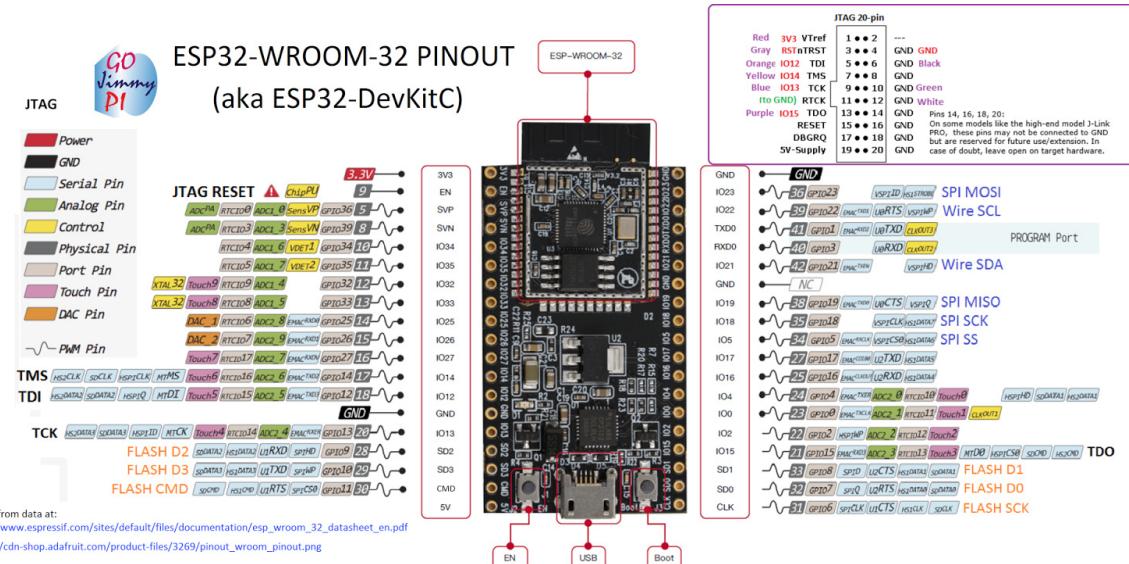
PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Pinout Diagram*
 - *Debug Tools*
 - *On-Board Debug Tools*
 - *External Debug Tools*

Pinout Diagram

JTAG Wiring Connections

Board Pin	JTAG Tool Pin
IO13	TCK
IO12	TDI
IO15	TDO
IO14	TMS
EN	RST
GND	GND



Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Balls listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre- quency	Flash	RAM
esp-wrover-kit	Feifei-Kit sif ESP- WROVER- KIT	<i>FTDI Chip</i> (default, on-board), <i>Mini-Module FT2232H</i> , <i>Olimex ARM-USB-OCD-H</i> , <i>Olimex ARM-USB-OCD</i> , <i>Olimex ARM-USB-TINY-H</i> , <i>Olimex ARM-USB-TINY</i>	ESP32	2240MHz	4MB	320KB

External Debug Tools

Balls listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU	Frequency	Flash	RAM
alksesp32	ALKS ESP32	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
esp32-evb	OLIMEX ESP32-EVB	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
esp32-gate	OLIMEX ESP32- GATEWAY	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
esp32dev	Espressif ESP32 Dev Module	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
esp32doit	DOIT iESP32 DEVKIT V1	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
esp32thing	SparkFun ESP32 Thing	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
esp32vn-i	ESP32n IoT Uno	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
espectro32	ESPECTRO32	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
espino32	ESPINO32	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
featheresp32	Aadafruit ESP32 Feather	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
firebeetle	Beetle- ESP32	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
hornbill132	Hornbill ESP32 Dev	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
hornbill132	Hornbill ESP32 Min- ima	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
lolin32	WEMOS LOLIN32	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
lolin_d32	WEMOS LOLIN D32	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
lolin_d32	WEMOS LOLIN D32 PRO	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
mhetesp32d	MHEt LIVE ESP32DevKIT	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
mhetesp32m	MHEt LIVE	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	
248	ESP32MiniKit	<i>Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	Chapter 1. Contents			
node32s	Node32s	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32240MHz	4MB	320KB	

Stable and upstream versions

You can switch between stable releases of Espressif 32 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = espressif32
board = ...

; Custom stable version
[env:custom_stable]
platform = espressif32@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-espressif32.git
board = ...
```

Packages

Name	Description
framework-arduinoespressif32	Arduino Wiring-based Framework (ESP32 Core)
framework-espifdf	Espressif IoT Development Framework
framework-pumbaa	Pumbaa Framework
framework-simba	Simba Framework
tool-espotapy	ESP8266 OTA utility
tool-esptoolpy	ESP8266 and ESP32 serial bootloader utility
tool-mkspiffs	Tool to build and unpack SPIFFS images
tool-openocd-esp32	OpenOCD for Espressif 32
toolchain-xtensa32	xtensa32-gcc

Warning: Linux Users:

- Install “udev” rules *99-platformio-udev.rules*
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
<i>Arduin</i>	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.
<i>ESP-IDF</i>	Espressif IoT Development Framework. Official development framework for ESP32.
<i>Pumba</i>	Pumba is Python on top of Simba. The implementation is a port of MicroPython, designed for embedded devices with limited amount of RAM and code memory.
<i>Simba</i>	Simba is an RTOS and build framework. It aims to make embedded programming easy and portable.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

Adafruit

ID	Name	Debug	MCU	Frequency	Flash	RAM
featheresp32	Adafruit ESP32 Feather	Yes	ESP32	240MHz	4MB	320KB

Aiyarafun

ID	Name	Debug	MCU	Frequency	Flash	RAM
node32s	Node32s	Yes	ESP32	240MHz	4MB	320KB

April Brother

ID	Name	Debug	MCU	Frequency	Flash	RAM
espea32	April Brother ESPeA32	No	ESP32	240MHz	4MB	320KB

DFRobot

ID	Name	Debug	MCU	Frequency	Flash	RAM
firebeetle32	FireBeetle-ESP32	Yes	ESP32	240MHz	4MB	320KB

DOIT

ID	Name	Debug	MCU	Frequency	Flash	RAM
esp32doit-devkit-v1	DOIT ESP32 DEVKIT V1	Yes	ESP32	240MHz	4MB	320KB

Dongsen Technology

ID	Name	Debug	MCU	Frequency	Flash	RAM
pocket_32	Dongsen Tech Pocket 32	Yes	ESP32	240MHz	4MB	320KB

DycodeX

ID	Name	Debug	MCU	Frequency	Flash	RAM
espectro32	ESPectro32	Yes	ESP32	240MHz	4MB	320KB

ESP32vn

ID	Name	Debug	MCU	Frequency	Flash	RAM
esp32vn-iot-uno	ESP32vn IoT Uno	Yes	ESP32	240MHz	4MB	320KB

Electronic SweetPeas

ID	Name	Debug	MCU	Frequency	Flash	RAM
esp320	Electronic SweetPeas ESP320	No	ESP32	240MHz	4MB	320KB

Espressif

ID	Name	Debug	MCU	Frequency	Flash	RAM
esp-wrover-kit	Espressif ESP-WROVER-KIT	Yes	ESP32	240MHz	4MB	320KB
esp32dev	Espressif ESP32 Dev Module	Yes	ESP32	240MHz	4MB	320KB
pico32	ESP32 Pico Kit	No	ESP32	240MHz	4MB	320KB

Hardkernel

ID	Name	Debug	MCU	Frequency	Flash	RAM
odroid_esp32	ODROID-GO	No	ESP32	240MHz	16MB	320KB

Heltec Automation

ID	Name	Debug	MCU	Frequency	Flash	RAM
heltec_wifi_kit_32	Heltec WIFI Kit 32	No	ESP32	240MHz	4MB	320KB
heltec_wifi_lora_32	Heltec WIFI LoRa 32	No	ESP32	240MHz	4MB	320KB

Hornbill

ID	Name	Debug	MCU	Frequency	Flash	RAM
hornbill132dev	Hornbill ESP32 Dev	Yes	ESP32	240MHz	4MB	320KB
hornbill132minima	Hornbill ESP32 Minima	Yes	ESP32	240MHz	4MB	320KB

IntoRobot

ID	Name	Debug	MCU	Frequency	Flash	RAM
intorobot	IntoRobot Fig	No	ESP32	240MHz	4MB	320KB

M5Stack

ID	Name	Debug	MCU	Frequency	Flash	RAM
m5stack-core-esp32	M5Stack Core ESP32	No	ESP32	240MHz	4MB	320KB
m5stack-fire	M5Stack FIRE	No	ESP32	240MHz	16MB	320KB

MH-ET Live

ID	Name	Debug	MCU	Frequency	Flash	RAM
mhetesp32devkit	MH ET LIVE ESP32DevKIT	Yes	ESP32	240MHz	4MB	320KB
mhetesp32minikit	MH ET LIVE ESP32MiniKit	Yes	ESP32	240MHz	4MB	320KB

MakerAsia

ID	Name	Debug	MCU	Frequency	Flash	RAM
nano32	MakerAsia Nano32	No	ESP32	240MHz	4MB	320KB

Microduino

ID	Name	Debug	MCU	Frequency	Flash	RAM
microduino-core-esp32	Microduino Core ESP32	No	ESP32	240MHz	4MB	320KB

NodeMCU

ID	Name	Debug	MCU	Frequency	Flash	RAM
nodemcu-32s	NodeMCU-32S	Yes	ESP32	240MHz	4MB	320KB

Noduino

ID	Name	Debug	MCU	Frequency	Flash	RAM
quantum	Noduino Quantum	No	ESP32	240MHz	16MB	320KB

OLIMEX

ID	Name	Debug	MCU	Frequency	Flash	RAM
esp32-evb	OLIMEX ESP32-EVB	Yes	ESP32	240MHz	4MB	320KB
esp32-gateway	OLIMEX ESP32-GATEWAY	Yes	ESP32	240MHz	4MB	320KB

Onehorse

ID	Name	Debug	MCU	Frequency	Flash	RAM
onehorse32dev	Onehorse ESP32 Dev Module	No	ESP32	240MHz	4MB	320KB

RoboticsBrno

ID	Name	Debug	MCU	Frequency	Flash	RAM
alksesp32	ALKS ESP32	Yes	ESP32	240MHz	4MB	320KB

SparkFun Electronics

ID	Name	Debug	MCU	Frequency	Flash	RAM
esp32thing	SparkFun ESP32 Thing	Yes	ESP32	240MHz	4MB	320KB

TTGO

ID	Name	Debug	MCU	Frequency	Flash	RAM
ttgo-lora32-v1	TTGO LoRa32-OLED V1	Yes	ESP32	240MHz	4MB	320KB

ThaiEasyElec

ID	Name	Debug	MCU	Frequency	Flash	RAM
espino32	ESPino32	<i>Yes</i>	ESP32	240MHz	4MB	320KB

WEMOS

ID	Name	Debug	MCU	Frequency	Flash	RAM
lolin32	WEMOS LOLIN32	<i>Yes</i>	ESP32	240MHz	4MB	320KB
lolin_d32	WEMOS LOLIN D32	<i>Yes</i>	ESP32	240MHz	4MB	320KB
lolin_d32_pro	WEMOS LOLIN D32 PRO	<i>Yes</i>	ESP32	240MHz	4MB	320KB
wemosbat	WeMos WiFi & Bluetooth Battery	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Widora

ID	Name	Debug	MCU	Frequency	Flash	RAM
widora-air	Widora AIR	No	ESP32	240MHz	16MB	320KB

XinaBox

ID	Name	Debug	MCU	Frequency	Flash	RAM
xinabox_cw02	XinaBox CW02	<i>Yes</i>	ESP32	240MHz	4MB	320KB

u-blox

ID	Name	Debug	MCU	Frequency	Flash	RAM
nina_w10	u-blox NINA-W10 series	No	ESP32	240MHz	2MB	320KB

Espressif 8266

platform = espressif8266

Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

For more detailed information please visit [vendor site](#).

Contents

- [*Configuration*](#)
- [*Examples*](#)
- [*Stable and upstream versions*](#)

- *Packages*
- *Frameworks*
- *Boards*

Configuration

- *CPU Frequency*
- *FLASH Frequency*
- *FLASH Mode*
- *Reset Method*
- *Flash Size*
- *Upload Speed*
- *lwIP Variant*
- *Serial Debug*
- *Debug Level*
- *VTables*
- *Uploading files to file system SPIFFS*
- *Over-the-Air (OTA) update*
 - *Authentication and upload options*
- *Demo*
- *Using Arduino Framework with Staging version*

CPU Frequency

See `board_build.f_cpu` option from *Project Configuration File platformio.ini*

```
[env:myenv]
; set frequency to 160MHz
board_build.f_cpu = 160000000L
```

FLASH Frequency

Please use `board_build.f_flash` option from *Project Configuration File platformio.ini* to change a value. Possible values:

- 20000000L
- 26000000L
- 40000000L (default)
- 80000000L

```
[env:myenv]
; set frequency to 80MHz
board_build.f_flash = 80000000L
```

FLASH Mode

Flash chip interface mode. This parameter is stored in the binary image header, along with the flash size and flash frequency. The ROM bootloader in the ESP chip uses the value of these parameters in order to know how to talk to the flash chip.

Please use `board_build.flash_mode` option from *Project Configuration File platformio.ini* to change a value. Possible values:

- qio
- qout
- dio
- dout

```
[env:myenv]
board_build.flash_mode = qio
```

Reset Method

You can set custom reset method using `upload_resetmethod` option from *Project Configuration File platformio.ini*.

The possible values are:

- ck - RTS controls RESET or CH_PD, DTR controls GPIO0
- wifi0 - TXD controls GPIO0 via PNP transistor and DTR controls RESET via a capacitor
- nodemcu - GPIO0 and RESET controlled using two NPN transistors as in NodeMCU devkit.

See default reset methods per board.

```
[env:myenv]
upload_resetmethod = ck
```

Flash Size

Warning: Please make sure to read [ESP8266 Flash layout](#) information first.

Available LD-scripts: <https://github.com/esp8266/Arduino/tree/master/tools/sdk/ld>

- eagle.flash.512k0.1d 512K (no SPIFFS)
- eagle.flash.512k64.1d 512K (64K SPIFFS)
- eagle.flash.512k128.1d 512K (128K SPIFFS)
- eagle.flash.1m0.1d 1M (no SPIFFS)
- eagle.flash.1m64.1d 1M (64K SPIFFS)

- eagle.flash.1m128.ld 1M (128K SPIFFS)
- eagle.flash.1m144.ld 1M (144K SPIFFS)
- eagle.flash.1m160.ld 1M (160K SPIFFS)
- eagle.flash.1m192.ld 1M (192K SPIFFS)
- eagle.flash.1m256.ld 1M (256K SPIFFS)
- eagle.flash.1m512.ld 1M (512K SPIFFS)
- eagle.flash.2m.ld 2M (1M SPIFFS)
- eagle.flash.4m1m.ld 4M (1M SPIFFS)
- eagle.flash.4m2m.ld 4M (2M SPIFFS)
- eagle.flash.4m.ld 4M (3M SPIFFS)
- eagle.flash.8m.ld 8M (7M SPIFFS)
- eagle.flash.16m.ld 16M (15M SPIFFS)

To override default LD script please use `build_flags` from *Project Configuration File platformio.ini*.

```
[env:myenv]
build_flags = -Wl,-Teagle.flash.4m.ld
```

Upload Speed

You can set custom upload speed using `upload_speed` option from *Project Configuration File platformio.ini*

```
[env:myenv]
upload_speed = 9600
```

lwIP Variant

Available variants (macros):

- -D PIO_FRAMEWORK_ARDUINO_LWIP_HIGHER_BANDWIDTH v1.4 Higher Bandwidth (default)
- -D PIO_FRAMEWORK_ARDUINO_LWIP2_LOW_MEMORY v2 Lower Memory
- -D PIO_FRAMEWORK_ARDUINO_LWIP2_HIGHER_BANDWIDTH v2 Higher Bandwidth

You can change lwIP Variant passing a custom macro using project `build_flags`.

For example, switch to lwIP v2 Lower Memory

```
[env:myenv]
...
build_flags = -D PIO_FRAMEWORK_ARDUINO_LWIP2_LOW_MEMORY
```

Serial Debug

Please use the next `build_flags` to enable Serial debug:

```
[env:myenv]
...
build_flags = -DDEBUG_ESP_PORT=Serial

; or for Serial1
build_flags = -DDEBUG_ESP_PORT=Serial1
```

Debug Level

Please use one of the next `build_flags` to change debug level. A `build_flags` option could be used only the one time per build environment. If you need to specify more flags, please separate them with a new line or space.

Also, please note that you will need to extend `build_flags` with `Serial Debug` macro. For example, `build_flags = -DDEBUG_ESP_PORT=Serial -DDEBUG_ESP_SSL`

Actual information is available in Arduino for ESP8266 Board Manifest. Please scroll to `generic.menu.DebugLevel` section.

```
[env:myenv]
platform = ...
board = ...
framework = arduino

;;;; Possible options ;;;;;;

; SSL
build_flags = -DDEBUG_ESP_SSL

; TLS_MEM
build_flags = -DDEBUG_ESP_TLS_MEM

; HTTP_CLIENT
build_flags = -DDEBUG_ESP_HTTP_CLIENT

; HTTP_SERVER
build_flags = -DDEBUG_ESP_HTTP_SERVER

; SSL+TLS_MEM
build_flags =
  -DDEBUG_ESP_SSL
  -DDEBUG_ESP_TLS_MEM

; SSL+HTTP_CLIENT
build_flags =
  -DDEBUG_ESP_SSL
  -DDEBUG_ESP_HTTP_CLIENT

; SSL+HTTP_SERVER
build_flags =
  -DDEBUG_ESP_SSL
  -DDEBUG_ESP_HTTP_SERVER

; TLS_MEM+HTTP_CLIENT
build_flags =
  -DDEBUG_ESP_TLS_MEM
  -DDEBUG_ESP_HTTP_CLIENT
```

(continues on next page)

(continued from previous page)

```

; TLS+HTTP_SERVER
build_flags =
-DDEBUG_ESP_TLS_MEM
-DDEBUG_ESP_HTTP_SERVER

; HTTP_CLIENT+HTTP_SERVER
build_flags =
-DDEBUG_ESP_HTTP_CLIENT
-DDEBUG_ESP_HTTP_SERVER

; SSL+TLS+HTTP_CLIENT
build_flags =
-DDEBUG_ESP_SSL
-DDEBUG_ESP_TLS_MEM
-DDEBUG_ESP_HTTP_CLIENT

; SSL+TLS+HTTP_SERVER
build_flags =
-DDEBUG_ESP_SSL
-DDEBUG_ESP_TLS_MEM
-DDEBUG_ESP_HTTP_SERVER

; SSL+HTTP_CLIENT+HTTP_SERVER
build_flags =
-DDEBUG_ESP_SSL
-DDEBUG_ESP_HTTP_CLIENT
-DDEBUG_ESP_HTTP_SERVER

; TLS+HTTP_CLIENT+HTTP_SERVER
build_flags =
-DDEBUG_ESP_TLS_MEM
-DDEBUG_ESP_HTTP_CLIENT
-DDEBUG_ESP_HTTP_SERVER

; SSL+TLS+HTTP_CLIENT+HTTP_SERVER
build_flags =
-DDEBUG_ESP_SSL
-DDEBUG_ESP_TLS_MEM
-DDEBUG_ESP_HTTP_CLIENT
-DDEBUG_ESP_HTTP_SERVER

; CORE
build_flags = -DDEBUG_ESP_CORE

; WIFI
build_flags = -DDEBUG_ESP_WIFI

; HTTP_UPDATE
build_flags = -DDEBUG_ESP_HTTP_UPDATE

; UPDATER
build_flags = -DDEBUG_ESP_UPDATER

; OTA
build_flags = -DDEBUG_ESP_OTA

```

(continues on next page)

(continued from previous page)

```

; OOM
build_flags =
-DDEBUG_ESP_OOM
-include "umm_malloc/umm_malloc_cfg.h"

; CORE+WIFI+HTTP_UPDATE+UPDATER+OTA+OOM
build_flags =
-DDEBUG_ESP_CORE
-DDEBUG_ESP_WIFI
-DDEBUG_ESP_HTTP_UPDATE
-DDEBUG_ESP_UPDATER
-DDEBUG_ESP_OTA
-DDEBUG_ESP_OOM -include "umm_malloc/umm_malloc_cfg.h"

; SSL+TLS_MEM+HTTP_CLIENT+HTTP_SERVER+CORE+WIFI+HTTP_UPDATE+UPDATER+OTA+OOM
build_flags =
-DDEBUG_ESP_SSL
-DDEBUG_ESP_TLS_MEM
-DDEBUG_ESP_HTTP_CLIENT
-DDEBUG_ESP_HTTP_SERVER
-DDEBUG_ESP_CORE
-DDEBUG_ESP_WIFI
-DDEBUG_ESP_HTTP_UPDATE
-DDEBUG_ESP_UPDATER
-DDEBUG_ESP_OTA
-DDEBUG_ESP_OOM -include "umm_malloc/umm_malloc_cfg.h"

; NoAssert-NDEBUG
build_flags = -DNDEBUG

```

VTables

Please use one of the next *build_flags*:

```

[env:myenv]
...
; Flash (default)
build_flags = -DVTABLES_IN_FLASH

; Heap
build_flags = -DVTABLES_IN_DRAM

; IRAM
build_flags = -DVTABLES_IN_IRAM

```

Uploading files to file system SPIFFS

Warning: Please make sure to read [ESP8266 Flash layout](#) information first.

1. Initialize project `platformio init` (if you have not initialized yet)

2. Create data folder (it should be on the same level as `src` folder) and put files here. Also, you can specify own location for `data_dir`
3. Run `buildfs` or `uploadfs` target using `platformio run --target` command.

To upload SPIFFS image using OTA update please specify `upload_port` / `--upload-port` as IP address or mDNS host name (ending with the `*.local`). For the details please follow to [Over-the-Air \(OTA\) update](#).

By default, will be used default LD Script for the board where is specified SPIFFS offsets (start, end, page, block). You can override it using [Flash Size](#).

Active discussion is located in [issue #382](#).

Over-the-Air (OTA) update

Firstly, please read [What is OTA? How to use it?](#)

There are 2 options:

- Directly specify `platformio run --upload-port` in command line

```
platformio run --target upload --upload-port IP_ADDRESS_HERE or mDNS_NAME.local
```

- Specify `upload_port` option in *Project Configuration File platformio.ini*

```
[env:myenv]
upload_port = IP_ADDRESS_HERE or mDNS_NAME.local
```

For example,

- `platformio run -t upload --upload-port 192.168.0.255`
- `platformio run -t upload --upload-port myesp8266.local`

Authentication and upload options

You can pass additional options/flags to OTA uploader using `upload_flags` option in *Project Configuration File platformio.ini*

```
[env:myenv]
upload_flags = --port=8266
```

Available flags

- `--port=ESP_PORT` ESP8266 OTA Port. Default 8266
- `--auth=AUTH` Set authentication password
- `--spiffs` Use this option to transmit a SPIFFS image and do not flash the module

For the full list with available options please run

```
~/platformio/packages/tool-espotapy/espotapy.py -h

Usage: espotapy [options]

Transmit image over the air to the esp8266 module with OTA support.

Options:
```

(continues on next page)

(continued from previous page)

```
-h, --help           show this help message and exit

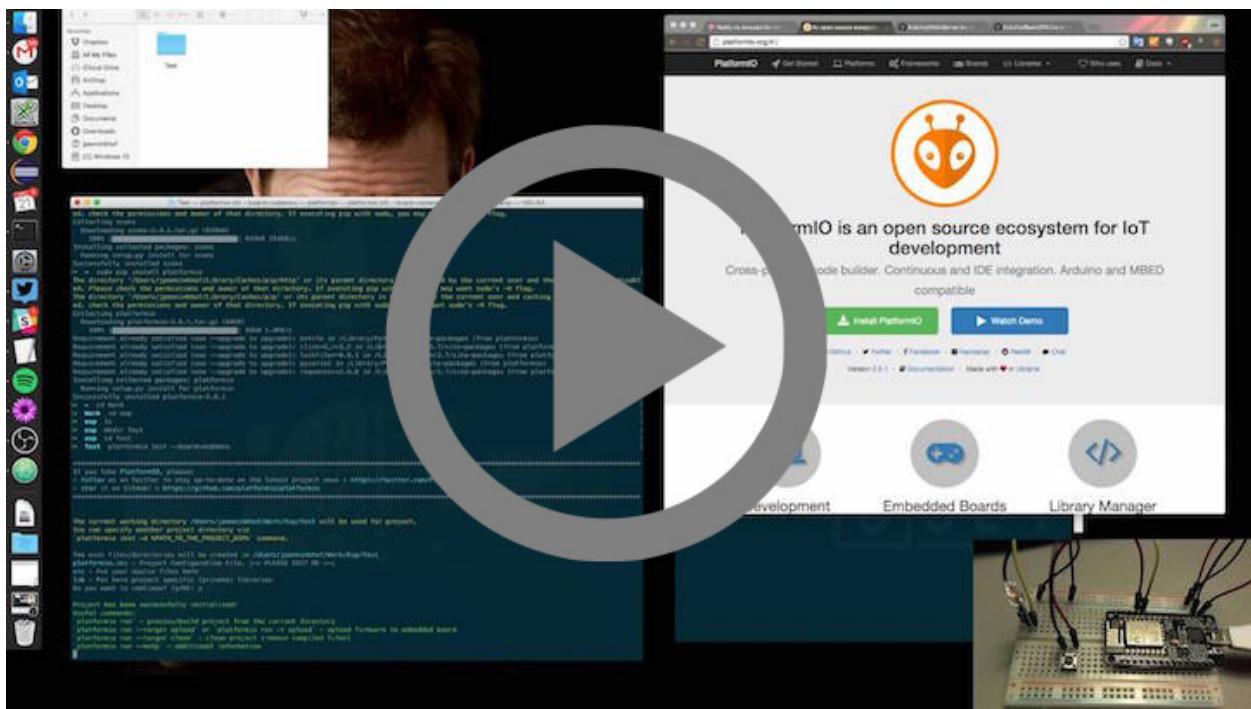
Destination:
  -i ESP_IP, --ip=ESP_IP
    ESP8266 IP Address.
  -p ESP_PORT, --port=ESP_PORT
    ESP8266 ota Port. Default 8266

Authentication:
  -a AUTH, --auth=AUTH
    Set authentication password.

Image:
  -f FILE, --file=FILE
    Image file.
  -s, --spiffs
    Use this option to transmit a SPIFFS image and do not
    flash the module.

Output:
  -d, --debug
    Show debug output. And override loglevel with debug.
  -r, --progress
    Show progress output. Does not work for ArduinoIDE
```

Demo



Using Arduino Framework with Staging version

PlatformIO will install the latest Arduino Core for ESP8266 from <https://github.com/esp8266/Arduino>. The [Git](#) should be installed in a system. To update Arduino Core to the latest revision, please open [PlatformIO IDE](#) and navigate to PIO Home > Platforms > Updates.

1. Please install *PlatformIO IDE*
2. Initialize a new project, open *Project Configuration File platformio.ini* and set *platform* to `https://github.com/platformio/platform-espressif8266.git#feature/stage`. For example,

```
[env:nodemcuv2]
platform = https://github.com/platformio/platform-espressif8266.git#feature/stage
board = nodemcuv2
framework = arduino
```

3. Try to build project
4. If you see build errors, then try to build this project using the same *stage* with Arduino IDE
5. If it works with Arduino IDE but doesn't work with PlatformIO, then please [file new issue](#) with attached information:
 - test project/files
 - detailed log of build process from Arduino IDE (please copy it from console to <https://hastebin.com>)
 - detailed log of build process from PlatformIO Build System (please copy it from console to <https://hastebin.com>)

Examples

Examples are listed from Espressif 8266 development platform repository:

- arduino-asyncudp
- arduino-blink
- arduino-webserver
- arduino-wifiscan
- esp8266-nonos-sdk-blink
- esp8266-rtos-sdk-blink
- native-sdk
- simba-blink

Stable and upstream versions

You can switch between stable releases of Espressif 8266 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = espressif8266
board = ...

; Custom stable version
[env:custom_stable]
```

(continues on next page)

(continued from previous page)

```
platform = espressif8266@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-espressif8266.git
board = ...
```

Packages

Name	Description
framework-arduinoespressif8266	Arduino Wiring-based Framework (ESP8266 Core)
framework-esp8266-nonos-sdk	ESP8266 Non-OS SDK
framework-esp8266-rtos-sdk	ESP8266 SDK based on FreeRTOS
framework-simba	Simba Framework
sdk-esp8266	ESP8266 SDK
tool-espotapy	ESP8266 OTA utility
tool-esptool	esptool-ck
tool-mkspiffs	Tool to build and unpack SPIFFS images
toolchain-xtensa	xtensa-gcc

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
<i>Arduino</i>	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.
<i>ESP8266 Non-OS SDK</i>	The non-OS SDK provides a set of application programming interfaces (APIs) for core ESP8266 functionalities such as data reception/transmission over Wi-Fi, TCP/IP stack functions, hardware interface functions and basic system management functions.
<i>ESP8266 RTOS SDK</i>	ESP8266 SDK based on FreeRTOS, a truly free professional grade RTOS for microcontrollers
<i>Simba</i>	Simba is an RTOS and build framework. It aims to make embedded programming easy and portable.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

4D Systems

ID	Name	Debug	MCU	Frequency	Flash	RAM
gen4iod	4D Systems gen4 IoD Range	No	ESP8266	80MHz	512KB	80KB

Adafruit

ID	Name	Debug	MCU	Frequency	Flash	RAM
huzzah	Adafruit HUZZAH ESP8266	No	ESP8266	80MHz	4MB	80KB

DigiStamp

ID	Name	Debug	MCU	Frequency	Flash	RAM
oak	DigiStamp Oak	No	ESP8266	80MHz	4MB	80KB

Doit

ID	Name	Debug	MCU	Frequency	Flash	RAM
espduino	ESPDuino (ESP-13 Module)	No	ESP8266	80MHz	4MB	80KB

DycodeX

ID	Name	Debug	MCU	Frequency	Flash	RAM
espectro	ESPectro Core	No	ESP8266	80MHz	4MB	80KB

ESPert

ID	Name	Debug	MCU	Frequency	Flash	RAM
espresso_lite_v1	ESPRESSO Lite 1.0	No	ESP8266	80MHz	4MB	80KB
espresso_lite_v2	ESPRESSO Lite 2.0	No	ESP8266	80MHz	4MB	80KB

ESPino

ID	Name	Debug	MCU	Frequency	Flash	RAM
espino	ESPino	No	ESP8266	80MHz	4MB	80KB

Espressif

ID	Name	Debug	MCU	Frequency	Flash	RAM
esp01	Espressif Generic ESP8266 ESP-01 512k	No	ESP8266	80MHz	512KB	80KB
esp01_1m	Espressif Generic ESP8266 ESP-01 1M	No	ESP8266	80MHz	1MB	80KB
esp07	Espressif Generic ESP8266 ESP-07	No	ESP8266	80MHz	4MB	80KB
esp12e	Espressif ESP8266 ESP-12E	No	ESP8266	80MHz	4MB	80KB
esp8285	Generic ESP8285 Module	No	ESP8266	80MHz	423.98KB	80KB
esp_wroom_02	ESP-WROOM-02	No	ESP8266	80MHz	2MB	80KB
phoenix_v1	Phoenix 1.0	No	ESP8266	80MHz	4MB	80KB
phoenix_v2	Phoenix 2.0	No	ESP8266	80MHz	4MB	80KB
wifinfo	WifInfo	No	ESP8266	80MHz	1MB	80KB

Heltec

ID	Name	Debug	MCU	Frequency	Flash	RAM
heltec_wifi_kit_8	Heltec Wifi kit 8	No	ESP8266	80MHz	4MB	80KB

NodeMCU

ID	Name	Debug	MCU	Frequency	Flash	RAM
nodemcu	NodeMCU 0.9 (ESP-12 Module)	No	ESP8266	80MHz	4MB	80KB
nodemcuv2	NodeMCU 1.0 (ESP-12E Module)	No	ESP8266	80MHz	4MB	80KB

Olimex

ID	Name	Debug	MCU	Frequency	Flash	RAM
modwifi	Olimex MOD-WIFI-ESP8266(-DEV)	No	ESP8266	80MHz	2MB	80KB

SeeedStudio

ID	Name	Debug	MCU	Frequency	Flash	RAM
wio_node	Wio Node	No	ESP8266	80MHz	4MB	80KB

SparkFun

ID	Name	Debug	MCU	Frequency	Flash	RAM
sparkfunBlynk	SparkFun Blynk Board	No	ESP8266	80MHz	4MB	80KB
thing	SparkFun ESP8266 Thing	No	ESP8266	80MHz	512KB	80KB
thingdev	SparkFun ESP8266 Thing Dev	No	ESP8266	80MHz	512KB	80KB

SweetPea

ID	Name	Debug	MCU	Frequency	Flash	RAM
esp210	SweetPea ESP-210	No	ESP8266	80MHz	4MB	80KB

ThaiEasyElec

ID	Name	Debug	MCU	Frequency	Flash	RAM
espinotee	ThaiEasyElec ESPino	No	ESP8266	80MHz	4MB	80KB

WEMOS

ID	Name	Debug	MCU	Frequency	Flash	RAM
d1	WEMOS D1 R1 (Retired)	No	ESP8266	80MHz	4MB	80KB
d1_mini	WeMos D1 R2 & mini	No	ESP8266	80MHz	4MB	80KB
d1_mini_lite	WeMos D1 mini Lite	No	ESP8266	80MHz	1MB	80KB
d1_mini_pro	WeMos D1 mini Pro	No	ESP8266	80MHz	16MB	80KB

Freescale Kinetis

platform = freescalekinetis

Freescale Kinetis Microcontrollers is family of multiple hardware- and software-compatible ARM Cortex-M0+, Cortex-M4 and Cortex-M7-based MCU series. Kinetis MCUs offer exceptional low-power performance, scalability and feature integration.

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Debugging*
- *Stable and upstream versions*
- *Packages*
- *Frameworks*
- *Boards*

Examples

Examples are listed from Freescale Kinetis development platform repository:

- [mbed-blink](#)
- [mbed-dsp](#)
- [mbed-events](#)
- [mbed-rtos](#)
- [mbed-rtos-ethernet](#)
- [mbed-serial](#)

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Bands listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Frequency	Flash	RAM
IBMEthernet	Ethernet IoT Starter Kit	CMSIS-DAP (on-board), J-LINK	MK64FN1M0VDC12	120MHz	1MB	256KB
frdm_k20d5	Freescale Kinetis FRDM-K20D50M	CMSIS-DAP (on-board), J-LINK	MK20DX128VLC128	48MHz	128KB	16KB
frdm_k22f	Freescale Kinetis FRDM-K22F	CMSIS-DAP (on-board), J-LINK	MK22FN512VLC128	48MHz	512KB	128KB
frdm_k64f	Freescale Kinetis FRDM-K64F	CMSIS-DAP (on-board), J-LINK	MK64FN1M0VDC12	120MHz	1MB	256KB
frdm_k66f	Freescale Kinetis FRDM-K66F	CMSIS-DAP (on-board), J-LINK	MK66FN2M0VDC12	120MHz	2MB	256KB
frdm_k105z	Freescale Kinetis FRDM-KL05Z	CMSIS-DAP (on-board), J-LINK	MKL05Z32VFLC12	48MHz	32KB	4KB
frdm_kl25z	Freescale Kinetis FRDM-KL25Z	CMSIS-DAP (on-board), Black Magic Probe, J-LINK	MKL25Z128VLC12	48MHz	128KB	16KB
frdm_kl27z	Freescale Kinetis FRDM-KL27Z	CMSIS-DAP (on-board), Black Magic Probe, J-LINK	MKL27Z64VLC12	48MHz	64KB	16KB
frdm_kl43z	Freescale Kinetis FRDM-KL43Z	CMSIS-DAP (on-board), J-LINK	MKL43Z256VLC12	48MHz	256KB	32KB
frdm_kl46z	Freescale Kinetis FRDM-KL46Z	CMSIS-DAP (on-board), J-LINK	MKL46Z256VLC12	48MHz	256KB	32KB
frdm_kw41z	Freescale Kinetis FRDM-KW41Z	CMSIS-DAP (on-board), J-LINK	MKW41Z512VLC12	48MHz	512KB	128KB

External Debug Tools

Boards listed below are compatible with [PIO Unified Debugger](#) but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU	Frequency	Flash	RAM
hexiwear	Hexiwear	CMSIS-DAP, J-LINK	MK64FN1M0VDC12	120MHz	1MB	256KB

Stable and upstream versions

You can switch between stable releases of Freescale Kinetis development platform and the latest upstream version using `platform` option in *Project Configuration File* `platformio.ini` as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = freescalekinetis
board = ...

; Custom stable version
[env:custom_stable]
platform = freescalekinetis@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-freescalekinetis.git
board = ...
```

Packages

Name	Description
framework-mbed	mbed Framework
tool-jlink	SEGGER J-Link Software and Documentation Pack
tool-pyocd	Open source python library for programming and debugging ARM Cortex-M microcontrollers using CMSIS-DAP
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

Freescale

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
IBMEthernetKit	Ethernet IoT Starter Kit	Yes	MK64FN1M0VLL12	120MHz	1MB	256KB
frdm_k20d50m	Freescale Kinetis FRDM-K20D50M	Yes	MK20DX128VLH5	48MHz	128KB	16KB
frdm_k22f	Freescale Kinetis FRDM-K22F	Yes	MK22FN512VLH12	120MHz	512KB	128KB
frdm_k64f	Freescale Kinetis FRDM-K64F	Yes	MK64FN1M0VLL12	120MHz	1MB	256KB
frdm_k66f	Freescale Kinetis FRDM-K66F	Yes	MK66FN2M0VMD18I	80MHz	2MB	256KB
frdm_kl05z	Freescale Kinetis FRDM-KL05Z	Yes	MKL05Z32VFM4	48MHz	32KB	4KB
frdm_kl25z	Freescale Kinetis FRDM-KL25Z	Yes	MKL25Z128VLK4	48MHz	128KB	16KB
frdm_kl27z	Freescale Kinetis FRDM-KL27Z	Yes	MKL27Z64VLH4	48MHz	64KB	16KB
frdm_kl43z	Freescale Kinetis FRDM-KL43Z	Yes	MKL43Z256VLH4	48MHz	256KB	32KB
frdm_kl46z	Freescale Kinetis FRDM-KL46Z	Yes	MKL46Z256VLL4	48MHz	256KB	32KB
frdm_kw41z	Freescale Kinetis FRDM-KW41Z	Yes	MKW41Z512VHT4	48MHz	512KB	128KB

MikroElektronika

ID	Name	Debug	MCU	Frequency	Flash	RAM
hexiwear	Hexiwear	Yes	MK64FN1M0VDC12	120MHz	1MB	256KB

Infineon XMC

`platform = infineonxmc`

Infineon has designed the XMC microcontrollers for real-time critical applications with an industry-standard core. The XMC microcontrollers can be integrated with the Arduino platform

For more detailed information please visit [vendor site](#).

Contents

- [Examples](#)
- [Debugging](#)
- [Stable and upstream versions](#)
- [Packages](#)
- [Frameworks](#)

- *Boards*

Examples

Examples are listed from Infineon XMC development platform repository:

- arduino-blink
- arduino-wire
- device-control
- ifx9201
- rtc
- spi
- ultrasonic

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Bands listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
xmc1100_boot_kit	XMC1100 Boot Kit	J-LINK (on-board)	XMC1100	32MHz	64KB	64KB
xmc1100_h_bridge2go	XMC1100 H-Bridge 2Go	J-LINK (on-board)	XMC1100	32MHz	64KB	64KB
xmc1100_xmc2go	XMC1100 XMC2Go	J-LINK (on-board)	XMC1100	32MHz	64KB	64KB
xmc1300_boot_kit	XMC1300 Boot Kit	J-LINK (on-board)	XMC1300	32MHz	64KB	64KB
xmc1300_sense2gol	XMC1300 Sense2GoL	J-LINK (on-board)	XMC1300	32MHz	64KB	122.23KB
xmc4200_distance2go	XMC4200 Distance2Go	J-LINK (on-board)	XMC4200	80MHz	250KB	256KB
xmc4700_relax_kit	XMC4700 Relax Kit	J-LINK (on-board)	XMC4700	144MHz	2.00MB	1.95MB

Stable and upstream versions

You can switch between stable releases of Infineon XMC development platform and the latest upstream version using `platform` option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = infineonxmc
board = ...

; Custom stable version
[env:custom_stable]
platform = infineonxmc@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/Infineon/platformio-infineonxmc.git
board = ...
```

Packages

Name	Description
framework-arduinomxmc	Arduino Wiring-based Framework (Infineon XMC Core)
tool-jlink	SEGGER J-Link Software and Documentation Pack
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Arduino	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

Infineon

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
xmc1100_boot_kit	XMC1100 Boot Kit	<i>Yes</i>	XMC1100	32MHz	64KB	64KB
xmc1100_h_bridge2go	XMC1100 H-Bridge 2Go	<i>Yes</i>	XMC1100	32MHz	64KB	64KB
xmc1100_xmc2go	XMC1100 XMC2Go	<i>Yes</i>	XMC1100	32MHz	64KB	64KB
xmc1300_boot_kit	XMC1300 Boot Kit	<i>Yes</i>	XMC1300	32MHz	64KB	64KB
xmc1300_sense2go_l	XMC1300 Sense2GoL	<i>Yes</i>	XMC1300	32MHz	64KB	122.23KB
xmc4200_distance2go	XMC4200 Distance2Go	<i>Yes</i>	XMC4200	80MHz	250KB	256KB
xmc4700_relax_kit	XMC4700 Relax Kit	<i>Yes</i>	XMC4700	144MHz	2.00MB	1.95MB

Intel ARC32

platform = intel_arc32

ARC embedded processors are a family of 32-bit CPUs that are widely used in SoC devices for storage, home, mobile, automotive, and Internet of Things applications.

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Stable and upstream versions*
- *Packages*
- *Frameworks*
- *Boards*

Examples

Examples are listed from Intel ARC32 development platform repository:

- arduino-blink
- arduino-curie-imu
- arduino-internal-libs

Stable and upstream versions

You can switch between stable releases of Intel ARC32 development platform and the latest upstream version using *platform* option in *Project Configuration File* *platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = intel_arc32
board = ...

; Custom stable version
[env:custom_stable]
platform = intel_arc32@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-intel_arc32.git
board = ...
```

Packages

Name	Description
framework-arduinointel	Arduino Wiring-based Framework (Intel ARC Core)
tool-arduino101load	Genuino101 uploader
toolchain-intelarc32	GCC for Intel ARC

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Arduin	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

Intel

ID	Name	Debug	MCU	Frequency	Flash	RAM
genuino101	Arduino/Genuino 101	No	ARCV2EM	32MHz	152KB	80KB

Lattice iCE40

[platform](#) = lattice_ice40

The iCE40 family of ultra-low power, non-volatile FPGAs has five devices with densities ranging from 384 to 7680 Look-Up Tables (LUTs). In addition to LUT-based, low-cost programmable logic, these devices feature Embedded Block RAM (EBR), Non-volatile Configuration Memory (NVCM) and Phase Locked Loops (PLLs). These features allow the devices to be used in low-cost, high-volume consumer and system applications.

For more detailed information please visit [vendor site](#).

Contents

- [Examples](#)
- [Stable and upstream versions](#)
- [Packages](#)
- [Boards](#)

Examples

Examples are listed from Lattice iCE40 development platform repository:

- counter
- leds

Stable and upstream versions

You can switch between stable releases of Lattice iCE40 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = lattice_ice40
board = ...

; Custom stable version
[env:custom_stable]
platform = lattice_ice40@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-lattice_ice40.git
board = ...
```

Packages

Name	Description
toolchain-icestorm	Tools for analyzing and creating bitstream files for FPGA IceStorm
toolchain-iverilog	Verilog simulation and synthesis tool

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

FPGAwars

ID	Name	Debug	MCU	Frequency	Flash	RAM
icezum	IceZUM Alhambra FPGA	No	ICE40-HX1K-TQ144	12MHz	32KB	32KB

Lattice

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
icestick	Lattice iCEstick FPGA Evaluation Kit	No	ICE40-HX1K-TQ144	12MHz	32KB	32KB

Maxim 32

platform = maxim32

Maxim's microcontrollers provide low-power, efficient, and secure solutions for challenging embedded applications. Maxim's processors embed cutting-edge technologies to secure data and intellectual property, proven analog circuitry for real-world applications, and battery-conserving low power operation.

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Debugging*
- *Stable and upstream versions*
- *Packages*

- *Frameworks*
- *Boards*

Examples

Examples are listed from [Maxim 32 development platform repository](#):

- [mbed-blink](#)
- [mbed-dsp](#)
- [mbed-events](#)
- [mbed-rtos](#)
- [mbed-serial](#)

Debugging

[PIO Unified Debugger](#) - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging [Tools & Debug Probes](#) using [debug_tool](#) options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Boards listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
max32600mb	Maxim ARM mbed Enabled Development Platform for MAX32600	CMSIS-DAP (on-board)	MAX32600	4MHz	256KB	32KB

External Debug Tools

Boards listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
maxwsnenv	Maxim Wireless Sensor Node Demonstrator	CMSIS-DAP	MAX32610	24MHz	256KB	32KB

Stable and upstream versions

You can switch between stable releases of Maxim 32 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = maxim32
board = ...

; Custom stable version
[env:custom_stable]
platform = maxim32@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-maxim32.git
board = ...
```

Packages

Name	Description
framework-mbed	mbed Framework
tool-pyocd	Open source python library for programming and debugging ARM Cortex-M microcontrollers using CMSIS-DAP
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules *99-platformio-udev.rules*
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

Boards**Note:**

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

Maxim

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
max32600mbed	Maxim ARM mbed Enabled Development Platform for MAX32600	Yes	MAX32600	24MHz	256KB	32KB
max32620hsp	Maxim Health Sensor Platform	No	MAX32620	96MHz	2MB	256KB
max32625mbed	MAX32625MBED	No	MAX32625	96MHz	512KB	160KB
max32625nexpaq	MAX32625NEXPAQ	No	MAX32625	96MHz	512KB	160KB
max32630fthr	Maxim MAX32630FTHR Application Platform	No	MAX32630	96MHz	2MB	512KB
maxwsnenv	Maxim Wireless Sensor Node Demonstrator	Yes	MAX3261	24MHz	256KB	32KB

Microchip PIC32

[platform](#) = microchippic32

Microchip's 32-bit portfolio with the MIPS microAptiv or M4K core offer high performance microcontrollers, and all the tools needed to develop your embedded projects. PIC32 MCUs gives your application the processing power, memory and peripherals your design needs!

For more detailed information please visit [vendor site](#).

Contents

- [Examples](#)

- *Stable and upstream versions*
- *Packages*
- *Frameworks*
- *Boards*

Examples

Examples are listed from Microchip PIC32 development platform repository:

- [arduino-blink](#)
- [arduino-internal-libs](#)

Stable and upstream versions

You can switch between stable releases of Microchip PIC32 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = microchippic32
board = ...

; Custom stable version
[env:custom_stable]
platform = microchippic32@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-microchippic32.git
board = ...
```

Packages

Name	Description
framework-arduinomicrochippic32	Arduino Wiring-based Framework (PIC32 Core)
tool-pic32prog	pic32prog
toolchain-microchippic32	GCC for Microchip PIC32

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Ar-duino	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

4DSystems

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
picadillo_35t	4DSystems PICadillo 35T	No	32MX795F512L	80MHz	508KB	128KB

Digilent

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
cerebot32mx4	Digilent Cerebot 32MX4	No	32MX460F512L	80MHz	508KB	32KB
cerebot32mx7	Digilent Cerebot 32MX7	No	32MX795F512L	80MHz	508KB	128KB
chipkit_cmod	Digilent chipKIT Cmod	No	32MX150F128D	40MHz	124KB	32KB
chipkit_dp32	Digilent chipKIT DP32	No	32MX250F128B	40MHz	120KB	32KB
chipkit_mx3	Digilent chipKIT MX3	No	32MX320F128H	80MHz	124KB	16KB
chipkit_pro_mx4	Digilent chipKIT Pro MX4	No	32MX460F512L	80MHz	508KB	32KB
chipkit_pro_mx7	Digilent chipKIT Pro MX7	No	32MX795F512L	80MHz	508KB	128KB
chipkit_uc32	Digilent chipKIT uC32	No	32MX340F512H	80MHz	508KB	32KB
chipkit_wf32	Digilent chipKIT WF32	No	32MX695F512L	80MHz	508KB	128KB
chipkit_wifire	Digilent chipKIT WiFire	No	32MZ2048ECG100	200MHz	1.98MB	512KB
mega_pic32	Digilent chipKIT MAX32	No	32MX795F512L	80MHz	508KB	128KB
openscope	Digilent OpenScope	No	32MZ2048EFG124	200MHz	1.98MB	512KB
uno_pic32	Digilent chipKIT UNO32	No	32MX320F128H	80MHz	124KB	16KB

Fubarino

ID	Name	Debug	MCU	Frequency	Flash	RAM
fubarino_mini	Fubarino Mini	No	32MX250F128D	48MHz	120KB	32KB
fubarino_sd	Fubarino SD (1.5)	No	32MX795F512H	80MHz	508KB	128KB

MikroElektronika

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
clicker2	MikroElektronika Clicker 2	No	32MX460F512L	80MHz	508KB	32KB
flipnclickmz	MikroElektronika Flip N Click MZ	No	32MZ2048EFH100	252MHz	1.98MB	512KB

Olimex

ID	Name	Debug	MCU	Frequency	Flash	RAM
pinguino32	Olimex PIC32-PINGUINO	No	32MX440F256H	80MHz	252KB	32KB

OpenBCI

ID	Name	Debug	MCU	Frequency	Flash	RAM
openbci	OpenBCI 32bit	No	32MX250F128B	40MHz	120KB	32KB

PONTECH

ID	Name	Debug	MCU	Frequency	Flash	RAM
usbono_pic32	PONTECH UAV100	No	32MX440F512H	80MHz	508KB	32KB

Pontech

ID	Name	Debug	MCU	Frequency	Flash	RAM
nofire	Pontech NoFire	No	32MZ2048EFG100	200MHz	1.98MB	512KB
quick240_usb	Pontech Quick240	No	32MX795F512L	80MHz	508KB	128KB

SeeedStudio

ID	Name	Debug	MCU	Frequency	Flash	RAM
cui32stem	SeeedStudio CUI32stem	No	32MX795F512H	80MHz	508KB	128KB

UBW32

ID	Name	Debug	MCU	Frequency	Flash	RAM
ubw32_mx460	UBW32 MX460	No	32MX460F512L	80MHz	508KB	32KB
ubw32_mx795	UBW32 MX795	No	32MX795F512L	80MHz	508KB	128KB

chipKIT

ID	Name	Debug	MCU	Frequency	Flash	RAM
lenny	chipKIT Lenny	No	32MX270F256D	40MHz	120KB	32KB

element14

ID	Name	Debug	MCU	Frequency	Flash	RAM
chipkit_pi	Element14 chipKIT Pi	No	32MX250F128B	40MHz	120KB	32KB

Nordic nRF51

platform = nordicnrf51

The Nordic nRF51 Series is a family of highly flexible, multi-protocol, system-on-chip (SoC) devices for ultra-low power wireless applications. nRF51 Series devices support a range of protocol stacks including Bluetooth Smart (previously called Bluetooth low energy), ANT and proprietary 2.4GHz protocols such as Gazell.

For more detailed information please visit [vendor site](#).

Contents

- [*Examples*](#)
- [*Debugging*](#)
- [*Stable and upstream versions*](#)
- [*Packages*](#)
- [*Frameworks*](#)
- [*Boards*](#)

Examples

Examples are listed from Nordic nRF51 development platform repository:

- [arduino-ble-led](#)
- [arduino-blink](#)
- [arduino-internal-libs](#)
- [mbed-ble-thermometer](#)
- [mbed-blink](#)
- [mbed-events](#)
- [mbed-microbit-hello-world](#)
- [mbed-serial](#)

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- [*Debug Tools*](#)
 - [*On-Board Debug Tools*](#)
 - [*External Debug Tools*](#)

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Banks listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
bbcmicrobit	BBC micro:bit	<i>CMSIS-DAP</i> (on-board)	NRF51820MHz	256KB16KB		
bbcmicrobit_B(S130)	BBC micro:bit B(S130)	<i>CMSIS-DAP</i> (on-board)	NRF51820MHz	256KB16KB		
delta_dfcm_nn50	Delta DFCM-NNN50	<i>CMSIS-DAP</i> (on-board)	NRF51820MHz	256KB16KB		
dfcm_nnn40	Delta DFCM-NNN40	<i>CMSIS-DAP</i> (on-board)	NRF51820MHz	256KB32KB		
hrm1017	Switch Science mbed HRM1017	<i>CMSIS-DAP</i> (on-board)	NRF51820MHz	256KB16KB		
nrf51822_y5_ybug	nRF51822 mbug	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe</i> , <i>J-LINK</i> , <i>ST-LINK</i>	NRF51820MHz	256KB16KB		
nrf51_dk	Nordic nRF51-DK	<i>CMSIS-DAP</i> (on-board), <i>J-LINK</i> (on-board), <i>Black Magic Probe</i> , <i>ST-LINK</i>	NRF51820MHz	256KB32KB		
nrf51_dongle	Nordic nRF51-Dongle	<i>CMSIS-DAP</i> (on-board), <i>J-LINK</i> (on-board)	NRF51820MHz	256KB32KB		
nrf51_mkit	Nordic nRF51822-mKIT	<i>CMSIS-DAP</i> (on-board)	NRF51820MHz	128KB16KB		
redBearLab	RedBearLab nRF51822	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe</i> , <i>J-LINK</i> , <i>ST-LINK</i>	NRF51820MHz	256KB16KB		
redBearLab_BT	RedBearLab BLE Nano 1.5	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe</i> , <i>J-LINK</i> , <i>ST-LINK</i>	NRF51820MHz	256KB32KB		
seeedArchBLE	Seeed Arch BLE	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe</i> , <i>J-LINK</i> , <i>ST-LINK</i>	NRF51820MHz	128KB16KB		
seeedArchLink	Seeed Arch Link	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe</i> , <i>J-LINK</i> , <i>ST-LINK</i>	NRF51820MHz	256KB16KB		
seeedTinyBLE	Seeed Tiny BLE	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe</i> , <i>J-LINK</i> , <i>ST-LINK</i>	NRF51820MHz	256KB16KB		
ty51822r3	Switch Science mbed TY51822r3	<i>CMSIS-DAP</i> (on-board)	NRF51820MHz	256KB32KB		
vbluno51	VNG VBLUNO51	<i>CMSIS-DAP</i> (on-board)	NRF51820MHz	128KB32KB		
wallbot_ble	JKSoft Wallbot BLE	<i>CMSIS-DAP</i> (on-board)	NRF51820MHz	128KB16KB		

External Debug Tools

Boards listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
bluz_dk	BluzDK	<i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF51822	32MHz	256KB	32KB
ng_beacon	ng-beacon	<i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF51822	32MHz	256KB	32KB
oshchip	OSHChip	<i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF51822	32MHz	256KB	32KB
rfdduino	RFduino	<i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF51822	16MHz	128KB	8KB
waveshare_ble400	Waveshare BLE400	<i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF51822	32MHz	256KB	32KB

Stable and upstream versions

You can switch between stable releases of Nordic nRF51 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = nordicnrf51
board = ...

; Custom stable version
[env:custom_stable]
platform = nordicnrf51@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-nordicnrf51.git
board = ...
```

Packages

Name	Description
framework-arduinoonordicnrf5	Arduino Wiring-based Framework (Nordic NRF5 Core)
framework-mbed	mbed Framework
tool-jlink	SEGGER J-Link Software and Documentation Pack
tool-nrfjprog	nRF5x command line tool
tool-openocd	OpenOCD
tool-sreccat	Merging tool
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Arduin	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

BBC

ID	Name	Debug	MCU	Frequency	Flash	RAM
bbcmicrobit	BBC micro:bit	Yes	NRF51822	16MHz	256KB	16KB
bbcmicrobit_b	BBC micro:bit B(S130)	Yes	NRF51822	16MHz	256KB	16KB

BluzDK

ID	Name	Debug	MCU	Frequency	Flash	RAM
bluz_dk	BluzDK	Yes	NRF51822	32MHz	256KB	32KB

Delta

ID	Name	Debug	MCU	Frequency	Flash	RAM
delta_dfcm_nnn50	Delta DFCM-NNN50	Yes	NRF51822	32MHz	256KB	16KB
dfcm_nnn40	Delta DFCM-NNN40	Yes	NRF51822	32MHz	256KB	32KB

JKSoft

ID	Name	Debug	MCU	Frequency	Flash	RAM
wallbot_ble	JKSoft Wallbot BLE	Yes	NRF51822	16MHz	128KB	16KB

Nordic

ID	Name	Debug	MCU	Frequency	Flash	RAM
nrf51_dk	Nordic nRF51-DK	Yes	NRF51822	32MHz	256KB	32KB
nrf51_dongle	Nordic nRF51-Dongle	Yes	NRF51822	32MHz	256KB	32KB
nrf51_mkit	Nordic nRF51822-mKIT	Yes	NRF51822	16MHz	128KB	16KB

OSHChip

ID	Name	Debug	MCU	Frequency	Flash	RAM
oshchip	OSHChip	Yes	NRF51822	32MHz	256KB	32KB

RFduino

ID	Name	Debug	MCU	Frequency	Flash	RAM
rfduino	RFduino	Yes	NRF51822	16MHz	128KB	8KB

RedBearLab

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
redBearLab	RedBearLab nRF51822	Yes	NRF51822	16MHz	256KB	16KB
redBearLabBLENano	RedBearLab BLE Nano 1.5	Yes	NRF51822	16MHz	256KB	32KB

SeeedStudio

ID	Name	Debug	MCU	Frequency	Flash	RAM
seeedArchBLE	Seeed Arch BLE	Yes	NRF51822	16MHz	128KB	16KB
seeedArchLink	Seeed Arch Link	Yes	NRF51822	16MHz	256KB	16KB
seeedTinyBLE	Seeed Tiny BLE	Yes	NRF51822	16MHz	256KB	16KB

Switch Science

ID	Name	Debug	MCU	Frequency	Flash	RAM
hrm1017	Switch Science mbed HRM1017	Yes	NRF51822	16MHz	256KB	16KB
ty51822r3	Switch Science mbed TY51822r3	Yes	NRF51822	32MHz	256KB	32KB

VNG

ID	Name	Debug	MCU	Frequency	Flash	RAM
vbluno51	VNG VBLUNO51	Yes	NRF51822	16MHz	128KB	32KB

Waveshare

ID	Name	Debug	MCU	Frequency	Flash	RAM
waveshare_ble400	Waveshare BLE400	Yes	NRF51822	32MHz	256KB	32KB

ng-beacon

ID	Name	Debug	MCU	Frequency	Flash	RAM
ng_beacon	ng-beacon	Yes	NRF51822	32MHz	256KB	32KB

y5 design

ID	Name	Debug	MCU	Frequency	Flash	RAM
nrf51822_y5_mbug	y5 nRF51822 mbug	Yes	NRF51822	16MHz	256KB	16KB

Nordic nRF52

platform = nordicnrf52

The nRF52 Series are built for speed to carry out increasingly complex tasks in the shortest possible time and return to sleep, conserving precious battery power. They have a Cortex-M4F processor and are the most capable Bluetooth Smart SoCs on the market.

For more detailed information please visit [vendor site](#).

Contents

- *Tutorials*
- *Examples*
- *Debugging*
- *Stable and upstream versions*
- *Packages*
- *Frameworks*
- *Boards*

Tutorials

- *Arduino and Nordic nRF52-DK: debugging and unit testing*

Examples

Examples are listed from Nordic nRF52 development platform repository:

- `arduino-ble-led`
- `arduino-blink`
- `mbed-ble-thermometer`
- `mbed-blink`
- `mbed-dsp`
- `mbed-events`
- `mbed-rtos`
- `mbed-serial`

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Banks listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
delta_dfbm_nQ620	DFBM-NQ620	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF52830	4MHz	512KB	64KB
nrf52840_dk	Nordic nRF52840-DK	<i>CMSIS-DAP</i> (on-board), <i>J-LINK</i> (on-board), <i>Black Magic Probe, ST-LINK</i>	NRF52840	4MHz	1MB	256KB
nrf52_dk	Nordic nRF52-DK	<i>CMSIS-DAP</i> (on-board), <i>J-LINK</i> (on-board), <i>Black Magic Probe, ST-LINK</i>	NRF52830	4MHz	512KB	64KB
redbear_blen	RnbdBearLab BLE Nano 2	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF52830	4MHz	512KB	64KB
redbear_blen	R2dBearLab Blend 2	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF52830	4MHz	512KB	64KB
ublox_evk_nina	inablob1 EVK-NINA-B1	<i>J-LINK</i> (on-board), <i>Black Magic Probe, ST-LINK</i>	NRF52830	4MHz	512KB	64KB

External Debug Tools

Banks listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
stct_nrf52_minidev	Tech Century nRF52 mini board	<i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF52830	4MHz	512KB	64KB

Stable and upstream versions

You can switch between stable releases of Nordic nRF52 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = nordicnrf52
board = ...

; Custom stable version
[env:custom_stable]
```

(continues on next page)

(continued from previous page)

```
platform = nordicnrf52@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-nordicnrf52.git
board = ...
```

Packages

Name	Description
framework-arduinonordicnrf5	Arduino Wiring-based Framework (Nordic NRF5 Core)
framework-mbed	mbed Framework
tool-jlink	SEGGER J-Link Software and Documentation Pack
tool-nrfjprog	nRF5x command line tool
tool-openocd	OpenOCD
tool-sreccat	Merging tool
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Arduinonordicnrf5	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

Delta

ID	Name	Debug	MCU	Frequency	Flash	RAM
delta_dfbm_nq620	Delta DFBM-NQ620	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

Nordic

ID	Name	Debug	MCU	Frequency	Flash	RAM
nrf52840_dk	Nordic nRF52840-DK	<i>Yes</i>	NRF52840	64MHz	1MB	256KB
nrf52_dk	Nordic nRF52-DK	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

RedBearLab

ID	Name	Debug	MCU	Frequency	Flash	RAM
redbear_blenano2	RedBearLab BLE Nano 2	<i>Yes</i>	NRF52832	64MHz	512KB	64KB
redbear_blend2	RedBearLab Blend 2	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

Taida Century

ID	Name	Debug	MCU	Frequency	Flash	RAM
stct_nrf52_minidev	Taida Century nRF52 mini board	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

u-blox

ID	Name	Debug	MCU	Frequency	Flash	RAM
ublox_evk_nina_b1	u-blox EVK-NINA-B1	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

NXP LPC

platform = nxplpc

The NXP LPC is a family of 32-bit microcontroller integrated circuits by NXP Semiconductors. The LPC chips are grouped into related series that are based around the same 32-bit ARM processor core, such as the Cortex-M4F, Cortex-M3, Cortex-M0+, or Cortex-M0. Internally, each microcontroller consists of the processor core, static RAM memory, flash memory, debugging interface, and various peripherals.

For more detailed information please visit [vendor site](#).

Contents

- [*Examples*](#)
- [*Debugging*](#)
- [*Stable and upstream versions*](#)
- [*Packages*](#)
- [*Frameworks*](#)
- [*Boards*](#)

Examples

Examples are listed from [NXP LPC development platform repository](#):

- [mbed-blink](#)
- [mbed-dsp](#)
- [mbed-events](#)
- [mbed-http-client](#)
- [mbed-rtos](#)
- [mbed-rtos-ethernet](#)
- [mbed-serial](#)

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- [*Debug Tools*](#)
 - [*On-Board Debug Tools*](#)
 - [*External Debug Tools*](#)

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging [*Tools & Debug Probes*](#) using [*debug_tool*](#) options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Boards listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
elektor_coc6nCoosi-Co!		<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC812	30MHz	16KB	4KB
lpc1114fn28	Switch Science mbed LPC1114FN28	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC1114FN28	248MHz	32KB	4KB
lpc11u24	NXP mbed LPC11U24	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC11U24	48MHz	32KB	8KB
lpc11u24_30	ARM mbed LPC11U24 (+CAN)	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC11U24	48MHz	32KB	8KB
lpc11u68	LPCXpresso11U68	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC11U68	50MHz	256KB	36KB
lpc1768	NXP mbed LPC1768	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC1768	96MHz	512KB	64KB
lpc4088	Embedded Artists LPC4088 QuickStart Board	<i>CMSIS-DAP</i> (on-board), <i>J-LINK</i>	LPC4088	120MHz	512KB	96KB
lpc4088_dm	Embedded Artists LPC4088 Display Module	<i>CMSIS-DAP</i> (on-board), <i>J-LINK</i>	LPC4088	120MHz	512KB	96KB
lpc4330_m4	Bambino-210E	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC4330	204MHz	8MB	264KB
lpc54114	NXP LPCXpresso54114	<i>CMSIS-DAP</i> (on-board), <i>J-LINK</i>	LPC54114J256BDM4	256BDM4Hz	256KB	192KB
lpc546xx	NXP LPCXpresso54608	<i>J-LINK</i> (on-board)	LPC54608ET	180MHz	512KB	200KB
lpc812	NXP LPC800-MAX	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC812	30MHz	16KB	4KB
lpc824	LPCXpresso824-MAX	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC824	30MHz	32KB	8KB
seeedArchPro	Seeed Arch Pro	<i>CMSIS-DAP</i> (on-board)	LPC1768	96MHz	512KB	64KB
ssci824	Switch Science mbed LPC824	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC824	30MHz	32KB	8KB
ubloxc027	u-blox C027	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK</i>	LPC1768	96MHz	512KB	64KB

External Debug Tools

Boards listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
blueboard_lpc11	NXP Technologies BlueBoard-LPC11U24	Black Magic Probe, J-LINK	LPC11U2448MHz	32KB	8KB	
dipcortexm0	Solder Splash Labs DipCortex M0	Black Magic Probe, J-LINK	LPC11U2450MHz	32KB	8KB	
lpc11c24	NXP LPC11C24	Black Magic Probe, J-LINK	LPC11C2448MHz	32KB	8KB	
lpc11u34_421	NXP LPC11U34	Black Magic Probe, J-LINK	LPC11U3448MHz	40KB	8KB	
lpc11u35	EA LPC11U35 QuickStart Board	Black Magic Probe, J-LINK	LPC11U3548MHz	64KB	10KB	
lpc11u35_501	CQ Publishing TG-LPC11U35-501	Black Magic Probe, J-LINK	LPC11U3548MHz	64KB	10KB	
lpc11u35_y5_mbug	NXP LPC11U35 mbug	Black Magic Probe, J-LINK	LPC11U3548MHz	64KB	10KB	
lpc11u37_501	NXP LPC11U37	Black Magic Probe, J-LINK	LPC11U3748MHz	128KB	10KB	
lpc1347	DipCortex M3	J-LINK	LPC1347	72MHz	64KB	12KB
lpc1549	NXP LPCXpresso1549	Black Magic Probe, J-LINK	LPC1549	72MHz	256KB	36KB

Stable and upstream versions

You can switch between stable releases of NXP LPC development platform and the latest upstream version using `platform` option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = nxplpc
board = ...

; Custom stable version
[env:custom_stable]
platform = nxplpc@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-nxplpc.git
board = ...
```

Packages

Name	Description
framework-mbed	mbed Framework
tool-jlink	SEGGER J-Link Software and Documentation Pack
tool-openocd	OpenOCD
tool-pyocd	Open source python library for programming and debugging ARM Cortex-M microcontrollers using CMSIS-DAP
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

AppNearMe

ID	Name	Debug	MCU	Frequency	Flash	RAM
micronfcboard	MicroNFCBoard	No	LPC11U34	48MHz	48KB	10KB

CQ Publishing

ID	Name	Debug	MCU	Frequency	Flash	RAM
lpc11u35_501	CQ Publishing TG-LPC11U35-501	Yes	LPC11U35	48MHz	64KB	10KB

Elektor Labs

ID	Name	Debug	MCU	Frequency	Flash	RAM
elektor_cocorico	CoCo-ri-Co!	Yes	LPC812	30MHz	16KB	4KB

Embedded Artists

ID	Name	Debug	MCU	Frequency	Flash	RAM
lpc11u35	EA LPC11U35 QuickStart Board	Yes	LPC11U35	48MHz	64KB	10KB
lpc4088	Embedded Artists LPC4088 QuickStart Board	Yes	LPC4088	120MHz	512KB	96KB
lpc4088_dm	Embedded Artists LPC4088 Display Module	Yes	LPC4088	120MHz	512KB	96KB

GHI Electronics

ID	Name	Debug	MCU	Frequency	Flash	RAM
oc_mbuino	mBuino	No	LPC11U24	50MHz	32KB	10KB

Micromint

ID	Name	Debug	MCU	Frequency	Flash	RAM
lpc4330_m4	Bambino-210E	Yes	LPC4330	204MHz	8MB	264KB

NGX Technologies

ID	Name	Debug	MCU	Frequency	Flash	RAM
blueboard_lpc11u2	NGX Technologies BlueBoard-LPC11U24	Yes	LPC11U24	48MHz	32KB	8KB

NXP

ID	Name	Debug	MCU	Frequency	Flash	RAM
lpc11c24	NXP LPC11C24	Yes	LPC11C24	48MHz	32KB	8KB
lpc11u24	NXP mbed LPC11U24	Yes	LPC11U24	48MHz	32KB	8KB
lpc11u24_301	ARM mbed LPC11U24 (+CAN)	Yes	LPC11U24	48MHz	32KB	8KB
lpc11u34_421	NXP LPC11U34	Yes	LPC11U34	48MHz	40KB	8KB
lpc11u37_501	NXP LPC11U37	Yes	LPC11U37	48MHz	128KB	10KB
lpc11u68	LPCXpresso11U68	Yes	LPC11U68	50MHz	256KB	36KB
lpc1549	NXP LPCXpresso1549	Yes	LPC1549	72MHz	256KB	36KB
lpc1768	NXP mbed LPC1768	Yes	LPC1768	96MHz	512KB	64KB
lpc54114	NXP LPCXpresso54114	Yes	LPC54114J256BD64	100MHz	256KB	192KB
lpc546xx	NXP LPCXpresso54608	Yes	LPC54608ET512	180MHz	512KB	200KB
lpc812	NXP LPC800-MAX	Yes	LPC812	30MHz	16KB	4KB
lpc824	LPCXpresso824-MAX	Yes	LPC824	30MHz	32KB	8KB

Outrageous Circuits

ID	Name	Debug	MCU	Frequency	Flash	RAM
mbuino	Outrageous Circuits mBuino	No	LPC11U24	48MHz	32KB	8KB

SeeedStudio

ID	Name	Debug	MCU	Frequency	Flash	RAM
seeedArchGPRS	Seeed Arch GPRS V2	No	LPC11U37	48MHz	128KB	10KB
seeedArchPro	Seeed Arch Pro	Yes	LPC1768	96MHz	512KB	64KB
xadow_m0	Seeed Xadow M0	No	LPC11U35	48MHz	64KB	10KB

Smeshlink

ID	Name	Debug	MCU	Frequency	Flash	RAM
xbed_lpc1768	Smeshlink xbed LPC1768	No	LPC1768	96MHz	512KB	32KB

Solder Splash Labs

ID	Name	Debug	MCU	Frequency	Flash	RAM
dipcortexm0	Solder Splash Labs DipCortex M0	Yes	LPC11U24	50MHz	32KB	8KB
lpc1347	DipCortex M3	Yes	LPC1347	72MHz	64KB	12KB

Switch Science

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
lpc1114fn28	Switch Science mbed LPC1114FN28	Yes	LPC1114FN28	48MHz	32KB	4KB
ssci824	Switch Science mbed LPC824	Yes	LPC824	30MHz	32KB	8KB

u-blox

ID	Name	Debug	MCU	Frequency	Flash	RAM
ubloxc027	u-blox C027	Yes	LPC1768	96MHz	512KB	64KB

y5 design

ID	Name	Debug	MCU	Frequency	Flash	RAM
lpc11u35_y5_mbug	y5 LPC11U35 mbug	Yes	LPC11U35	48MHz	64KB	10KB

RISC-V

platform = riscv

RISC-V is an open, free ISA enabling a new era of processor innovation through open standard collaboration. Born in academia and research, RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.

For more detailed information please visit [vendor site](#).

Contents

- [Examples](#)
- [Debugging](#)
- [Stable and upstream versions](#)
- [Packages](#)
- [Frameworks](#)
- [Boards](#)

Examples

Examples are listed from [RISC-V development platform repository](#):

- [freedom-e-sdk_asm](#)
- [freedom-e-sdk_dhrystone](#)

- freedom-e-sdk_gpio
- freedom-e-sdk_hello
- freedom-e-sdk_smp

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Bands listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Frequency	Flash	RAM
freedom-e300-hifive1	HiFive1	<i>FTDI Chip</i> (on-board)	FE310	320MHz	16MB	16KB

External Debug Tools

Bands listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU	Fre- quency	Flash	RAM
coreplexip-e31	Freedom E310 Arty (Artix-7) FPGA Dev Kit	<i>Olimex ARM-USB-TINY-H</i>	E31	320MHz	16MB	256MB
coreplexip-e51	E51 Arty (Artix-7) FPGA Dev Kit	<i>Olimex ARM-USB-TINY-H</i>	E51	1500MHz	16MB	256MB

Stable and upstream versions

You can switch between stable releases of RISC-V development platform and the latest upstream version using *platform* option in *Project Configuration File* *platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = riscv
board = ...

; Custom stable version
[env:custom_stable]
platform = riscv@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-riscv.git
board = ...
```

Packages

Name	Description
framework-freedom-e-sdk	Open Source Software for Developing on the SiFive Freedom E Platform
tool-openocd	OpenOCD
toolchain-riscv	GNU toolchain for RISC-V, including GCC

Warning: Linux Users:

- Install “udev” rules *99-platformio-udev.rules*
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
<i>Freedom E SDK</i>	Open Source Software for Developing on the SiFive Freedom E Platform

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

SiFive

ID	Name	Debug	MCU	Frequency	Flash	RAM
freedom-e300-hifive1	HiFive1	Yes	FE310	320MHz	16MB	16KB

Xilinx

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
coreplexip-e31-art	Freedom E310 Arty (Artix-7) FPGA Dev Kit	Yes	E31	320MHz	16MB	256MB
coreplexip-e51-art	E51 Arty (Artix-7) FPGA Dev Kit	Yes	E51	1500MHz	16MB	256MB

Samsung ARTIK

`platform = samsung_artik`

The Samsung ARTIK Smart IoT platform brings hardware modules and cloud services together, with built-in security and an ecosystem of tools and partners to speed up your time-to-market.

For more detailed information please visit [vendor site](#).

Contents

- [Configuration](#)
- [Examples](#)
- [Debugging](#)
- [Stable and upstream versions](#)
- [Packages](#)
- [Frameworks](#)
- [Boards](#)

Configuration

- Windows
- macOS
- Linux

If you are using the Samsung ARTIK platform on macOS or Linux, you need to perform the configuration steps detailed below to enable support for deployment and debugging.

Windows

Usually Windows Update Service will automatically install FTDI driver. You can choose an off-line installation and install [FTDI driver](#) manually.

After installation, you will have two FTDI devices. Please use [zadig](#) tool to change one FTDI device to a “libusb” compatible device.

macOS

First check that you have a FTDI compatible driver on your mac:

```
kextstat | grep FTDI
```

You should have `com.apple.driver.AppleUSBFTDI`.

1. Install [ARTIK FTDI Driver](#)
2. Reboot your system.

Linux

Create a new file named `/etc/udev/rules.d/51-artik053.rules` and add the following line:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="0403", ATTR{idProduct}=="6010", MODE="0660",  
GROUP="plugdev", SYMLINK+="artik053-%n"
```

Examples

Examples are listed from [Samsung ARTIK development platform repository](#):

- [artik_sdk](#)
- [blink_led_wifi](#)
- [hello](#)

Debugging

[PIO Unified Debugger](#) - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

External Debug Tools

Banks listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU	Frequency	Flash	RAM
artik_053	Samsung ARTIK053	<i>FTDI Chip</i> (default)	S5JT200	320MHz	8MB	1.25MB

Stable and upstream versions

You can switch between *stable releases* of Samsung ARTIK development platform and the latest upstream version using *platform* option in *Project Configuration File* *platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = samsung_artik
board = ...

; Custom stable version
[env:custom_stable]
platform = samsung_artik@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-samsung_artik.git
board = ...
```

Packages

Name	Description
framework-tizenrt	TizenRT RTOS with library
tool-artik-openocd	OpenOCD for ARTIK
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Tizen RT	Tizen RT is a lightweight RTOS-based platform to support low-end IoT devices

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

Samsung

ID	Name	Debug	MCU	Frequency	Flash	RAM
artik_053	Samsung ARTIK053	Yes	S5JT200	320MHz	8MB	1.25MB

Silicon Labs EFM32

`platform = siliconlabsefm32`

Silicon Labs EFM32 Gecko 32-bit microcontroller (MCU) family includes devices that offer flash memory configurations up to 256 kB, 32 kB of RAM and CPU speeds up to 48 MHz. Based on the powerful ARM Cortex-M core, the Gecko family features innovative low energy techniques, short wake-up time from energy saving modes and a wide selection of peripherals, making it ideal for battery operated applications and other systems requiring high performance and low-energy consumption.

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Debugging*
- *Stable and upstream versions*
- *Packages*
- *Frameworks*
- *Boards*

Examples

Examples are listed from [Silicon Labs EFM32 development platform repository](#):

- [mbed-blink](#)
- [mbed-dsp](#)
- [mbed-events](#)
- [mbed-serial](#)

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging [Tools & Debug Probes](#) using [debug_tool](#) options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Bands listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
efm32gg_stk	Silicon Labs EFM32GG-STK3700 (Giant Gecko)	<i>J-LINK</i> (on-board), <i>Black Magic Probe</i>	EFM32GG990	F48MHz	1MB	128KB
efm32hg_stk	Silicon Labs SLSTK3400A USB-enabled (Happy Gecko)	<i>J-LINK</i> (on-board), <i>Black Magic Probe</i>	EFM32HG322	F54MHz	64KB	8KB
efm32lg_stk	Silicon Labs EFM32LG-STK3600 (Leopard Gecko)	<i>J-LINK</i> (on-board), <i>Black Magic Probe</i>	EFM32LG990	F48MHz	256KB	32KB
efm32pg_stk	Silicon Labs SLSTK3401A (Pearl Gecko)	<i>J-LINK</i> (on-board), <i>Black Magic Probe</i>	EFM32PG1B200	F54MHz	256KB	32KB
efm32wg_stk	Silicon Labs EFM32WG-STK3800 (Wonder Gecko)	<i>J-LINK</i> (on-board), <i>Black Magic Probe</i>	EFM32WG990	F48MHz	256KB	32KB
efm32zg_stk	Silicon Labs EFM32ZG-STK3200 (Zero Gecko)	<i>J-LINK</i> (on-board), <i>Black Magic Probe</i>	EFM32ZG222	F24MHz	32KB	4KB

Stable and upstream versions

You can switch between stable releases of Silicon Labs EFM32 development platform and the latest upstream version using *platform* option in *Project Configuration File* *platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = siliconlabsefm32
board = ...

; Custom stable version
[env:custom_stable]
platform = siliconlabsefm32@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-siliconlabsefm32.git
board = ...
```

Packages

Name	Description
framework-mbed	mbed Framework
tool-openocd	OpenOCD
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

Boards**Note:**

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

Silicon Labs

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
efm32gg_stk3	Silicon Labs EFM32GG-STK3700 (Giant Gecko)	Yes	EFM32GG990F1024MHz	1MB	128KB	
efm32hg_stk3	Silicon Labs SLSTK3400A USB-enabled (Happy Gecko)	Yes	EFM32HG322F6424MHz	64KB	8KB	
efm32lg_stk3	Silicon Labs EFM32LG-STK3600 (Leopard Gecko)	Yes	EFM32LG990F25648MHz	256KB	32KB	
efm32pg_stk3	Silicon Labs SLSTK3401A (Pearl Gecko)	Yes	EFM32PG1B200F256MHz	256KB	32KB	
efm32wg_stk3	Silicon Labs EFM32WG-STK3800 (Wonder Gecko)	Yes	EFM32WG990F25648MHz	256KB	32KB	
efm32zg_stk3	Silicon Labs EFM32ZG-STK3200 (Zero Gecko)	Yes	EFM32ZG222F3224MHz	32KB	4KB	

ST STM32

platform = ststm32

The STM32 family of 32-bit Flash MCUs based on the ARM Cortex-M processor is designed to offer new degrees of freedom to MCU users. It offers a 32-bit product range that combines very high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.

For more detailed information please visit [vendor site](#).

Contents

- [*Tutorials*](#)
- [*Examples*](#)
- [*Debugging*](#)
- [*Stable and upstream versions*](#)
- [*Packages*](#)
- [*Frameworks*](#)
- [*Boards*](#)

Tutorials

- [*STM32Cube HAL and Nucleo-F401RE: debugging and unit testing*](#)

Examples

Examples are listed from ST STM32 development platform repository:

- arduino-blink
- arduino-external-libs
- arduino-internal-libs
- arduino-mxchip-azureiot
- arduino-mxchip-filesystem
- arduino-mxchip-sensors
- arduino-mxchip-wifiscan
- cmsis-blink
- libopencm3-1bitsy
- libopencm3-blink
- mbed-blink
- mbed-dsp
- mbed-events
- mbed-filesystem
- mbed-rtos
- mbed-rtos-ethernet

- mbed-rtos-ethernet-tls
- mbed-rtos-semaphore
- mbed-serial
- spl-blink
- stm32cube-hal-blink
- stm32cube-ll-blink

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Bands listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug
b96b_f446ve	96Boards B96B-F446VE	ST-LINK (default, on-board), Black Mag
cloud_jam	RushUp Cloud-JAM	ST-LINK (default, on-board), Black Mag
cloud_jam_14	RushUp Cloud-JAM L4	ST-LINK (default, on-board), Black Mag
disco_f030r8	ST STM32F0308DISCOVERY	ST-LINK (default, on-board), Black Mag
disco_f051r8	ST STM32F0DISCOVERY	ST-LINK (default, on-board), Black Mag
disco_f100rb	ST STM32VLDISCOVERY	ST-LINK (default, on-board), Black Mag
disco_f303vc	ST STM32F3DISCOVERY	ST-LINK (default, on-board), Black Mag
disco_f334c8	ST 32F3348DISCOVERY	ST-LINK (default, on-board), Black Mag
disco_f401vc	ST 32F401CDISCOVERY	ST-LINK (default, on-board), Black Mag
disco_f407vg	ST STM32F4DISCOVERY	ST-LINK (default, on-board), Black Mag
disco_f411ve	ST 32F411EDISCOVERY	ST-LINK (default, on-board), Black Mag
disco_f413zh	ST 32F413HDISCOVERY	ST-LINK (default, on-board), Black Mag
disco_f429zi	ST 32F429IDISCOVERY	ST-LINK (default, on-board), Black Mag

Table 3 – continued from previous page

ID	Name	Debug
disco_f469ni	ST 32F469IDISCOVERY	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
disco_f746ng	ST 32F746GDISCOVERY	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
disco_f769ni	ST 32F769IDISCOVERY	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
disco_1053c8	ST 32L0538DISCOVERY	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
disco_1072cz_lrwan1	ST DISCO-L072CZ-LRWAN1	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
disco_l1152rb	ST STM32LDISCOVERY	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
disco_1475vg_iot01a	ST DISCO-L475VG-IOT01A	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
disco_1476vg	ST 32L476GDISCOVERY	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
eval_1073z	ST STM32L073Z-EVAL	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
mbed_connect_odin	Mbed Connect Cloud	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe</i>
mxchip_az3166	Microsoft Azure IoT Development Kit (MXChip AZ3166)	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f030r8	ST Nucleo F030R8	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f031k6	ST Nucleo F031K6	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f042k6	ST Nucleo F042K6	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f070rb	ST Nucleo F070RB	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f072rb	ST Nucleo F072RB	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f091rc	ST Nucleo F091RC	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f103rb	ST Nucleo F103RB	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f207zg	ST Nucleo F207ZG	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f302r8	ST Nucleo F302R8	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f303k8	ST Nucleo F303K8	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f303re	ST Nucleo F303RE	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f303ze	ST Nucleo F303ZE	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f334r8	ST Nucleo F334R8	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f401re	ST Nucleo F401RE	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f410rb	ST Nucleo F410RB	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f411re	ST Nucleo F411RE	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f412zg	ST Nucleo F412ZG	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f413zh	ST Nucleo F413ZH	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f429zi	ST Nucleo F429ZI	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f446re	ST Nucleo F446RE	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f446ze	ST Nucleo F446ZE	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f746zg	ST Nucleo F746ZG	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_f767zi	ST Nucleo F767ZI	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_1011k4	ST Nucleo L011K4	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_1031k6	ST Nucleo L031K6	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_1053r8	ST Nucleo L053R8	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_1073rz	ST Nucleo L073RZ	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_1152re	ST Nucleo L152RE	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_1432kc	ST Nucleo L432KC	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_1476rg	ST Nucleo L476RG	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
nucleo_1496zg	ST Nucleo L496ZG	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>
seeedArchMax	Seeed Arch Max	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i>

External Debug Tools

Banks listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU	Fre- quency	Flash	RAM
1bitsy_stm32	f4B1ngt	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F415RIG8MHz	1MB	128KB	
armstrap_eag	lAr024ap Eagle 1024	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F417VIG8MHz	1MB	192KB	
armstrap_eag	lAr048ap Eagle 2048	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F427VIG8MHz	1.99MB	256KB	
armstrap_eag	lAr512trap Eagle 512	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F407VIG8MHz	512KB	192KB	
bluepill_f103	BluePill F103C8	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103C8MHz	64KB	20KB	
elmo_f411re	Espotel LoRa Module	<i>Black Magic Probe, J-LINK, ST-LINK (default)</i>	STM32F411REMHz	512KB	128KB	
genericSTM32	F103C8 (20k RAM. 64k Flash)	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103C8MHz	64KB	20KB	
genericSTM32	F103CB (20k RAM. 128k Flash)	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103CBMHz	128KB	20KB	
genericSTM32	F103R8 (20k RAM. 64 Flash)	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103R8MHz	64KB	20KB	
genericSTM32	F103RB (20k RAM. 128k Flash)	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103RBMHz	128KB	20KB	
genericSTM32	F103RC (48k RAM. 256k Flash)	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103RCMHz	256KB	48KB	
genericSTM32	F103RE (64k RAM. 512k Flash)	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103REMHz	512KB	64KB	
genericSTM32	F103VC (48k RAM. 256k Flash)	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103VCMHz	256KB	48KB	
genericSTM32	F103VE (64k RAM. 512k Flash)	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103VEMHz	512KB	64KB	
genericSTM32	F407VE (192k RAM. 512k Flash)	<i>ST-LINK</i>	STM32F407VEM8MHz	502.23KB	28KB	
maple	Maple	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103REMHz	108KB	17KB	
maple_mini_b2	Maple Mini Boot-loader 2.0	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103CBMHz	120KB	20KB	
maple_mini_o	Maple Mini Original	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103CBMHz	108KB	17KB	
mote_1152rc	NAMote72	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32L152RQMHz	256KB	32KB	
mtb_ublox_o	in-blox ODIN-W2	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F439ZI68MHz	2MB	256KB	
mts_dragonfly	MTS Dragonfly	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F411REMHz	512KB	128KB	
mts_mdot_f405	MultiTech mDot	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F411REMHz	512KB	128KB	
mts_mdot_f411	MultiTech mDot F411	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F411REMHz	512KB	128KB	
ublox_c030_n2	ublox C030-N211 IoT Starter Kit	<i>Black Magic Probe, CMSIS-DAP, J-LINK, ST-LINK</i>	STM32F437V80MHz	1MB	256KB	
ublox_c030_u2	ublox C030-U201 IoT Starter Kit	<i>Black Magic Probe, CMSIS-DAP, J-LINK, ST-LINK</i>	STM32F437V80MHz	1MB	256KB	
ublox_evk_odi	n-blox EVK-ODIN-W2	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F439ZI68MHz	2MB	256KB	
1.10. Development Platforms	Multitech xDot	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32L151COMHz	256KB	32KB	315

Stable and upstream versions

You can switch between stable releases of ST STM32 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = ststm32
board = ...

; Custom stable version
[env:custom_stable]
platform = ststm32@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-ststm32.git
board = ...
```

Packages

Name	Description
framework-arduinostm32mxchip	Arduino Wiring-based Framework (ST STM32 MXChip Core)
framework-arduinoststm32	Arduino Wiring-based Framework (STM32 Core)
framework-cmsis	Vendor-independent hardware abstraction layer for the Cortex-M processor series
framework-libopencm3	libOpenCM3 Framework
framework-mbed	mbed Framework
framework-spl	Standard Peripheral Library for STM32 MCUs
framework-stm32cube	STM32Cube embedded software libraries
tool-openocd	OpenOCD
tool-stlink	ST-Link
tool-stm32duino	STM32Duino Tools
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules *99-platformio-udev.rules*
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Arduino	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.
CMSIS	The ARM Cortex Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for the Cortex-M processor series and specifies debugger interfaces. The CMSIS enables consistent and simple software interfaces to the processor for interface peripherals, real-time operating systems, and middleware. It simplifies software re-use, reducing the learning curve for new microcontroller developers and cutting the time-to-market for devices.
libOpenCM3	The libOpenCM3 framework aims to create a free/libre/open-source firmware library for various ARM Cortex-M0(+)/M3/M4 microcontrollers, including ST STM32, TI Tiva and Stellaris, NXP LPC 11xx, 13xx, 15xx, 17xx parts, Atmel SAM3, Energy Micro EFM32 and others.
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.
SPL	The ST Standard Peripheral Library provides a set of functions for handling the peripherals on the STM32 Cortex-M3 family. The idea is to save the user (the new user, in particular) having to deal directly with the registers.
STM32	STM32Cube embedded software libraries, including: The HAL hardware abstraction layer, enabling portability between different STM32 devices via standardized API calls; The Low-Layer (LL) APIs, a light-weight, optimized, expert oriented set of APIs designed for both performance and runtime efficiency.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

1BitSquared

ID	Name	Debug	MCU	Frequency	Flash	RAM
1bitsy_stm32f415rgt	1Bitsy	Yes	STM32F415RGT	168MHz	1MB	128KB

96Boards

ID	Name	Debug	MCU	Frequency	Flash	RAM
b96b_f446ve	96Boards B96B-F446VE	Yes	STM32F446VET6	168MHz	512KB	128KB

Armstrap

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
armstrap_eagle1024	Armstrap Eagle 1024	Yes	STM32F417VGT6	168MHz	1MB	192KB
armstrap_eagle2048	Armstrap Eagle 2048	Yes	STM32F427VIT6	168MHz	1.99MB	256KB
armstrap_eagle512	Armstrap Eagle 512	Yes	STM32F407VET6	168MHz	512KB	192KB

Espotel

ID	Name	Debug	MCU	Frequency	Flash	RAM
elmo_f411re	Espotel LoRa Module	Yes	STM32F411RET6	100MHz	512KB	128KB

Generic

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
bluepill_f103c8	BluePill F103C8	Yes	STM32F103C8T62MHz	64KB	20KB	
genericSTM32F103C8	STM32F103C8 (20k RAM. 64k Flash)	Yes	STM32F103C8T62MHz	64KB	20KB	
genericSTM32F103CB	STM32F103CB (20k RAM. 128k Flash)	Yes	STM32F103CBT62MHz	128KB	20KB	
genericSTM32F103R8	STM32F103R8 (20k RAM. 64 Flash)	Yes	STM32F103R8T62MHz	64KB	20KB	
genericSTM32F103RB	STM32F103RB (20k RAM. 128k Flash)	Yes	STM32F103RBT62MHz	128KB	20KB	
genericSTM32F103RC	STM32F103RC (48k RAM. 256k Flash)	Yes	STM32F103RCT62MHz	256KB	48KB	
genericSTM32F103RE	STM32F103RE (64k RAM. 512k Flash)	Yes	STM32F103RET62MHz	512KB	64KB	
genericSTM32F103VC	STM32F103VC (48k RAM. 256k Flash)	Yes	STM32F103VCT62MHz	256KB	48KB	
genericSTM32F103VE	STM32F103VE (64k RAM. 512k Flash)	Yes	STM32F103VET62MHz	512KB	64KB	
genericSTM32F407VE	STM32F407VE (192k RAM. 512k Flash)	Yes	STM32F407VET68MHz	502.23KB	128KB	

LeafLabs

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
maple	Maple	Yes	STM32F103RBT6	72MHz	108KB	17KB
maple_mini_b20	Maple Mini Bootloader 2.0	Yes	STM32F103CBT6	72MHz	120KB	20KB
maple_mini_origin	Maple Mini Original	Yes	STM32F103CBT6	72MHz	108KB	17KB

MXChip

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
mxchip_az316	Microsoft Azure IoT Development Kit (MXChip AZ3166)	Yes	STM32F412ZG	100MHz	1MB	256KB

MultiTech

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
mts_dragonfly_f411re	MTS Dragonfly	Yes	STM32F411RET6	100MHz	512KB	128KB
mts_mdot_f405rg	MultiTech mDot	Yes	STM32F411RET6	100MHz	512KB	128KB
mts_mdot_f411re	MultiTech mDot F411	Yes	STM32F411RET6	100MHz	512KB	128KB
xdot_l151cc	MultiTech xDot	Yes	STM32L151CCU6	32MHz	256KB	32KB

RushUp

ID	Name	Debug	MCU	Frequency	Flash	RAM
cloud_jam	RushUp Cloud-JAM	Yes	STM32F401RET6	84MHz	512KB	96KB
cloud_jam_14	RushUp Cloud-JAM L4	Yes	STM32L476RG	80MHz	1MB	128KB

ST

ID	Name	Debug	MCU	Frequency	Flash	RAM
disco_f030r8	ST STM32F0308DISCOVERY	Yes	STM32F030R8T6	48MHz	64KB	8KB
disco_f051r8	ST STM32F0DISCOVERY	Yes	STM32F051R8T6	48MHz	64KB	8KB
disco_f100rb	ST STM32VLDISCOVERY	Yes	STM32F100RBT6	24MHz	128KB	8KB
disco_f303vc	ST STM32F3DISCOVERY	Yes	STM32F303VCT6	72MHz	256KB	48KB
disco_f334c8	ST 32F3348DISCOVERY	Yes	STM32F334C8T6	72MHz	64KB	12KB
disco_f401vc	ST 32F401CDISCOVERY	Yes	STM32F401VCT6	84MHz	256KB	64KB
disco_f407vg	ST STM32F4DISCOVERY	Yes	STM32F407VGT6	168MHz	1MB	128KB
disco_f411ve	ST 32F411EDISCOVERY	Yes	STM32F411VET6	100MHz	512KB	128KB

Continued on next page

Table 4 – continued from previous page

ID	Name	Debug	MCU	Frequency	Flash	RAM
disco_f413zh	ST 32F413HDISCOVERY	Yes	STM32F413ZHT6	100MHz	512KB	128KB
disco_f429zi	ST 32F429IDISCOVERY	Yes	STM32F429ZIT6	180MHz	2MB	256KB
disco_f469ni	ST 32F469IDISCOVERY	Yes	STM32F469NIH6	180MHz	1MB	384KB
disco_f746ng	ST 32F746GDISCOVERY	Yes	STM32F746NGH6	216MHz	1MB	320KB
disco_f769ni	ST 32F769IDISCOVERY	Yes	STM32F769NIH6	80MHz	1MB	512KB
disco_1053c8	ST 32L0538DISCOVERY	Yes	STM32L053C8T6	32MHz	64KB	8KB
disco_1072cz_lrwan1	ST DISCO-L072CZ-LRWAN1	Yes	STM32L072CZ	32MHz	192KB	20KB
disco_1152rb	ST STM32LDISCOVERY	Yes	STM32L152RBT6	32MHz	128KB	16KB
disco_1475vg_iot01a	ST DISCO-L475VG-IOT01A	Yes	STM32L475VGT6	80MHz	1MB	128KB
disco_1476vg	ST 32L476GDISCOVERY	Yes	STM32L476VGT6	80MHz	1MB	128KB
eval_1073z	ST STM32L073Z-EVAL	Yes	STM32L073VZT6	32MHz	192KB	20KB
nucleo_f030r8	ST Nucleo F030R8	Yes	STM32F030R8T6	48MHz	64KB	8KB
nucleo_f031k6	ST Nucleo F031K6	Yes	STM32F031K6T6	48MHz	32KB	4KB
nucleo_f042k6	ST Nucleo F042K6	Yes	STM32F042K6T6	48MHz	32KB	6KB
nucleo_f070rb	ST Nucleo F070RB	Yes	STM32F070RBT6	48MHz	128KB	16KB
nucleo_f072rb	ST Nucleo F072RB	Yes	STM32F072RBT6	48MHz	128KB	16KB
nucleo_f091rc	ST Nucleo F091RC	Yes	STM32F091RCT6	48MHz	256KB	32KB
nucleo_f103rb	ST Nucleo F103RB	Yes	STM32F103RBT6	72MHz	128KB	20KB
nucleo_f207zg	ST Nucleo F207ZG	Yes	STM32F207ZGT6	120MHz	1MB	128KB
nucleo_f302r8	ST Nucleo F302R8	Yes	STM32F302R8T6	72MHz	64KB	16KB
nucleo_f303k8	ST Nucleo F303K8	Yes	STM32F303K8T6	72MHz	64KB	16KB
nucleo_f303re	ST Nucleo F303RE	Yes	STM32F303RET6	72MHz	512KB	64KB
nucleo_f303ze	ST Nucleo F303ZE	Yes	STM32F303ZET6	72MHz	512KB	64KB
nucleo_f334r8	ST Nucleo F334R8	Yes	STM32F334R8T6	72MHz	64KB	16KB
nucleo_f401re	ST Nucleo F401RE	Yes	STM32F401RET6	84MHz	512KB	96KB
nucleo_f410rb	ST Nucleo F410RB	Yes	STM32F410RBT6	100MHz	128KB	32KB
nucleo_f411re	ST Nucleo F411RE	Yes	STM32F411RET6	100MHz	512KB	128KB
nucleo_f412zg	ST Nucleo F412ZG	Yes	STM32F412ZGT6	100MHz	1MB	256KB
nucleo_f413zh	ST Nucleo F413ZH	Yes	STM32F413ZHT6	100MHz	512KB	128KB
nucleo_f429zi	ST Nucleo F429ZI	Yes	STM32F429ZIT6	180MHz	2MB	256KB
nucleo_f446re	ST Nucleo F446RE	Yes	STM32F446RET6	180MHz	512KB	128KB
nucleo_f446ze	ST Nucleo F446ZE	Yes	STM32F446ZET6	180MHz	512KB	128KB
nucleo_f746zg	ST Nucleo F746ZG	Yes	STM32F746ZGT6	216MHz	1MB	320KB
nucleo_f767zi	ST Nucleo F767ZI	Yes	STM32F767ZIT6	216MHz	2MB	512KB
nucleo_1011k4	ST Nucleo L011K4	Yes	STM32L011K4T6	32MHz	16KB	2KB
nucleo_1031k6	ST Nucleo L031K6	Yes	STM32L031K6T6	32MHz	32KB	8KB
nucleo_1053r8	ST Nucleo L053R8	Yes	STM32L053R8T6	32MHz	64KB	8KB
nucleo_1073rz	ST Nucleo L073RZ	Yes	STM32L073RZ	32MHz	192KB	20KB
nucleo_1152re	ST Nucleo L152RE	Yes	STM32L152RET6	32MHz	512KB	80KB
nucleo_1432kc	ST Nucleo L432KC	Yes	STM32L432KCU6	80MHz	256KB	64KB
nucleo_1476rg	ST Nucleo L476RG	Yes	STM32L476RGT6	80MHz	1MB	128KB
nucleo_1496zg	ST Nucleo L496ZG	Yes	STM32L496ZGT6	80MHz	1MB	128KB

SeeedStudio

ID	Name	Debug	MCU	Frequency	Flash	RAM
seeedArchMax	Seeed Arch Max	Yes	STM32F407VET6	168MHz	512KB	192KB

Semtech

ID	Name	Debug	MCU	Frequency	Flash	RAM
mote_1152rc	NAMote72	Yes	STM32L152RC	32MHz	256KB	32KB

u-blox

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
mbed_connect_odin	Mbed Connect Cloud	Yes	STM32F439ZIY6	168MHz	2MB	256KB
mtb_ublox_odin_w2	u-blox ODIN-W2	Yes	STM32F439ZIY6	168MHz	2MB	256KB
ublox_c030_n211	u-blox C030-N211 IoT Starter Kit	Yes	STM32F437VG	180MHz	1MB	256KB
ublox_c030_u201	u-blox C030-U201 IoT Starter Kit	Yes	STM32F437VG	180MHz	1MB	256KB
ublox_evk_odin_w2	u-blox EVK-ODIN-W2	Yes	STM32F439ZIY6	168MHz	2MB	256KB

Teensy

`platform = teensy`

Teensy is a complete USB-based microcontroller development system, in a very small footprint, capable of implementing many types of projects. All programming is done via the USB port. No special programmer is needed, only a standard USB cable and a PC or Macintosh with a USB port.

For more detailed information please visit [vendor site](#).

Contents

- *Configuration*
- *Examples*
- *Debugging*
- *Stable and upstream versions*
- *Packages*
- *Frameworks*
- *Boards*

Configuration

- *USB Features*

USB Features

If you want to use Teensy USB Features, you need to add special macro/define using *build_flags*:

- -D USB_SERIAL
- -D USB_KEYBOARDONLY
- -D USB_TOUCHSCREEN
- -D USB_HID_TOUCHSCREEN
- -D USB_HID
- -D USB_SERIAL_HID
- -D USB_MIDI
- -D USB_MIDI4
- -D USB_MIDI16
- -D USB_MIDI_SERIAL
- -D USB_MIDI4_SERIAL
- -D USB_MIDI16_SERIAL
- -D USB_AUDIO
- -D USB_MIDI_AUDIO_SERIAL
- -D USB_MIDI16_AUDIO_SERIAL
- -D USB_MTPDISK
- -D USB_RAWHID
- -D USB_FLIGHTSIM
- -D USB_FLIGHTSIM_JOYSTICK
- -D USB_EVERYTHING
- -D USB_DISABLED

A default macro is set to -D USB_SERIAL if no one is specified.

Example:

```
[env:teensy_hid_device]
platform = teensy
framework = arduino
board = teensy20
build_flags = -D USB_RAWHID
```

See [Teensy USB Examples](#).

Examples

Examples are listed from Teensy development platform repository:

- [arduino-blink](#)
- [arduino-hid-usb-mouse](#)

- arduino-internal-libs
- mbed-blink
- mbed-dsp
- mbed-events
- mbed-rtos
- mbed-serial

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

External Debug Tools

Bands listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Debug	MCU	Frequency	Flash	RAM
teensy31	Teensy 3.1 / 3.2	J-LINK	MK20DX256	72MHz	256KB	64KB
teensy35	Teensy 3.5	J-LINK	MK64FX512	120MHz	512KB	192KB
teensy36	Teensy 3.6	J-LINK	MK66FX1M0	180MHz	1MB	256KB
teensylc	Teensy LC	J-LINK	MKL26Z64	48MHz	62KB	8KB

Stable and upstream versions

You can switch between stable releases of Teensy development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = teensy
board = ...

; Custom stable version
[env:custom_stable]
platform = teensy@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-teensy.git
board = ...
```

Packages

Name	Description
framework-arduinoteensy	Arduino Wiring-based Framework
framework-mbed	mbed Framework
tool-jlink	SEGGER J-Link Software and Documentation Pack
tool-teensy	Teensy Loader
toolchain-atmelavr	avr-gcc
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Teensy programming uses only Windows built-in HID drivers. When Teensy is programmed to act as a USB Serial device, Windows XP, Vista, 7 and 8 require [this serial driver](#) is needed to access the COM port your program uses. No special driver installation is necessary on Windows 10.

Frameworks

Name	Description
Arduinoteensy	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

Teensy

ID	Name	Debug	MCU	Frequency	Flash	RAM
teensy20	Teensy 2.0	No	ATMEGA32U4	16MHz	31.50KB	2.50KB
teensy20pp	Teensy++ 2.0	No	AT90USB1286	16MHz	127KB	8KB
teensy30	Teensy 3.0	No	MK20DX128	48MHz	128KB	16KB
teensy31	Teensy 3.1 / 3.2	Yes	MK20DX256	72MHz	256KB	64KB
teensy35	Teensy 3.5	Yes	MK64FX512	120MHz	512KB	192KB
teensy36	Teensy 3.6	Yes	MK66FX1M0	180MHz	1MB	256KB
teensylc	Teensy LC	Yes	MKL26Z64	48MHz	62KB	8KB

TI MSP430

`platform = timsdp430`

MSP430 microcontrollers (MCUs) from Texas Instruments (TI) are 16-bit, RISC-based, mixed-signal processors designed for ultra-low power. These MCUs offer the lowest power consumption and the perfect mix of integrated peripherals for thousands of applications.

For more detailed information please visit [vendor site](#).

Contents

- [*Examples*](#)
- [*Debugging*](#)
- [*Stable and upstream versions*](#)
- [*Packages*](#)
- [*Frameworks*](#)
- [*Boards*](#)

Examples

Examples are listed from [TI MSP430 development platform repository](#):

- [arduino-blink](#)
- [energia-blink](#)
- [energia-internal-libs](#)

- native-blink

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Boards listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre- quency	Flash	RAM
1pmmsp430f5529	TI LaunchPad MSP-EXP430F5529LP	MSP- <i>MSP Debug</i> (on-board)	MSP430F5529	16MHz	128KB	8KB
1pmmsp430fr4138	TI LaunchPad MSP-EXP430FR4138LP	MSP- <i>MSP Debug</i> (on-board)	MSP430FR4138	8MHz	15KB	2KB
1pmmsp430fr5739	TI FraunchPad MSP-EXP430FR5739LP	MSP- <i>MSP Debug</i> (on-board)	MSP430FR5739	6MHz	16KB	512B
1pmmsp430fr5969	TI LaunchPad MSP-EXP430FR5969LP	MSP- <i>MSP Debug</i> (on-board)	MSP430FR5969	8MHz	64KB	2KB
1pmmsp430fr6989	TI LaunchPad MSP-EXP430FR6989LP	MSP- <i>MSP Debug</i> (on-board)	MSP430FR6989	8MHz	127KB	2KB
1pmmsp430g2553	TI LaunchPad MSP-EXP430G2553LP	MSP- <i>MSP Debug</i> (on-board)	MSP430G2553	16MHz	16KB	512B

Stable and upstream versions

You can switch between stable releases of TI MSP430 development platform and the latest upstream version using *platform* option in *Project Configuration File* *platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = timsp430
board = ...

; Custom stable version
[env:custom_stable]
platform = timsp430@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-timsp430.git
board = ...
```

Packages

Name	Description
framework-arduinomsp430	Arduino Wiring-based Framework (MSP430 Core)
framework-energiamsp430	Energia Wiring-based Framework (MSP430 Core)
tool-mspdebug	MSPDebug
toolchain-timsp430	msp-gcc

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Arduin	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.
Energ	Energia Wiring-based framework enables pretty much anyone to start easily creating microcontroller-based projects and applications. Its easy-to-use libraries and functions provide developers of all experience levels to start blinking LEDs, buzzing buzzers and sensing sensors more quickly than ever before.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

PanStamp

ID	Name	Debug	MCU	Frequency	Flash	RAM
panStampNRG	PanStamp NRG 1.1	No	CC430F5137	12MHz	31.88KB	4KB

TI

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
1pmfsp430f5529	TI LaunchPad MSP-EXP430F5529LP	Yes	MSP430F5529	16MHz	128KB	8KB
1pmfsp430fr4133	TI LaunchPad MSP-EXP430FR4133LP	Yes	MSP430FR4133	8MHz	15KB	2KB
1pmfsp430fr5739	TI FraunchPad MSP-EXP430FR5739LP	Yes	MSP430FR5739	16MHz	16KB	512B
1pmfsp430fr5969	TI LaunchPad MSP-EXP430FR5969LP	Yes	MSP430FR5969	8MHz	64KB	2KB
1pmfsp430fr6989	TI LaunchPad MSP-EXP430FR6989LP	Yes	MSP430FR6989	8MHz	127KB	2KB
1pmfsp430g2553	TI LaunchPad MSP-EXP430G2553LP	Yes	MSP430G2553	16MHz	16KB	512B

TI TIVA

`platform = titiva`

Texas Instruments TM4C12x MCUs offer the industry's most popular ARM Cortex-M4 core with scalable memory and package options, unparalleled connectivity peripherals, advanced application functions, industry-leading analog integration, and extensive software solutions.

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Debugging*
- *Stable and upstream versions*
- *Packages*

- *Frameworks*
- *Boards*

Examples

Examples are listed from [TI TIVA development platform repository](#):

- [energia-blink](#)
- [energia-internal-libs](#)
- [libopencm3-blink](#)
- [native-blink](#)

Debugging

[PIO Unified Debugger](#) - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging [Tools & Debug Probes](#) using [debug_tool](#) options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Boards listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
lplm4f120h5qr	TI LaunchPad (Stellaris) w/ lm4f120 (80MHz)	TI-ICDI (on-board)	LPLM4F120H5QR	80MHz	256KB	32KB
lptm4c1230c3pr	TI LaunchPad (Tiva C) w/ tm4c123 (80MHz)	TI-ICDI (on-board)	LPTM4C1230C3P	80MHz	256KB	32KB
lptm4c1294ncp5dt	TI LaunchPad (Tiva C) w/ tm4c129 (120MHz)	TI-ICDI (on-board)	LPTM4C1294NCPI	120MHz	1MB	256KB

Stable and upstream versions

You can switch between stable releases of TI TIVA development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = titiva
board = ...

; Custom stable version
[env:custom_stable]
platform = titiva@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-titiva.git
board = ...
```

Packages

Name	Description
framework-energiatativa	Energia Wiring-based Framework (LM4F Core)
framework-libopencm3	libOpenCM3 Framework
tool-lm4flash	Flash Programmer
tool-openocd	OpenOCD
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules [99-platformio-udev.rules](#)
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
Energia	Energia Wiring-based framework enables pretty much anyone to start easily creating microcontroller-based projects and applications. Its easy-to-use libraries and functions provide developers of all experience levels to start blinking LEDs, buzzing buzzers and sensing sensors more quickly than ever before.
libOpenCM3	The libOpenCM3 framework aims to create a free/libre/open-source firmware library for various ARM Cortex-M0(+) / M3 / M4 microcontrollers, including ST STM32, TI Tiva and Stellaris, NXP LPC 11xx, 13xx, 15xx, 17xx parts, Atmel SAM3, Energy Micro EFM32 and others.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

TI

ID	Name	De- bug	MCU	Fre- quency	Flash	RAM
lplm4f120h5qr	TI LaunchPad (Stellaris) w/ lm4f120 (80MHz)	Yes	LPLM4F120H5QR	80MHz	256KB	32KB
lptm4c1230c3pm	TI LaunchPad (Tiva C) w/ tm4c123 (80MHz)	Yes	LPTM4C1230C3P	80MHz	256KB	32KB
lptm4c1294ncp	TI LaunchPad (Tiva C) w/ tm4c129 (120MHz)	Yes	LPTM4C1294NCPDT	120MHz	1MB	256KB

WIZNet W7500

`platform = wiznet7500`

The IOP (Internet Offload Processor) W7500 is the one-chip solution which integrates an ARM Cortex-M0, 128KB Flash and hardwired TCP/IP core for various embedded application platform especially requiring Internet of things

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Debugging*
- *Stable and upstream versions*
- *Packages*
- *Frameworks*

- *Boards*

Examples

Examples are listed from WIZNet W7500 development platform repository:

- mbed-blink
- mbed-dsp
- mbed-events
- mbed-rtos
- mbed-serial

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Bands listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Debug	MCU	Fre-quency	Flash	RAM
wizwiki_w7500	WIZwiki-W7500	<i>CMSIS-DAP</i> (on-board), <i>J-LINK</i>	WIZNET7500	48MHz	128KB	48KB
wizwiki_w7500e	WIZwiki-W7500ECO	<i>CMSIS-DAP</i> (on-board), <i>J-LINK</i>	WIZ-NET7500ECO	48MHz	128KB	48KB
wizwiki_w7500p	WIZwiki-W7500P	<i>CMSIS-DAP</i> (on-board), <i>J-LINK</i>	WIZ-NET7500P	48MHz	128KB	48KB

Stable and upstream versions

You can switch between stable releases of WIZNet W7500 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = wiznet7500
board = ...

; Custom stable version
[env:custom_stable]
platform = wiznet7500@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-wiznet7500.git
board = ...
```

Packages

Name	Description
framework-mbed	mbed Framework
tool-jlink	SEGGER J-Link Software and Documentation Pack
tool-pyocd	Open source python library for programming and debugging ARM Cortex-M microcontrollers using CMSIS-DAP
toolchain-gccarmnoneabi	gcc-arm-embedded

Warning: Linux Users:

- Install “udev” rules *99-platformio-udev.rules*
- Raspberry Pi users, please read this article [Enable serial port on Raspberry Pi](#).

Windows Users:

Please check that you have a correctly installed USB driver from board manufacturer

Frameworks

Name	Description
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

WIZNet

ID	Name	De-bug	MCU	Fre-quency	Flash	RAM
wizwiki_w7500	WIZwiki-W7500	<i>Yes</i>	WIZNET7500	48MHz	128KB	48KB
wizwiki_w7500eco	WIZwiki-W7500ECO	<i>Yes</i>	WIZ-NET7500ECO	48MHz	128KB	48KB
wizwiki_w7500p	WIZwiki-W7500P	<i>Yes</i>	WIZNET7500P	48MHz	128KB	48KB

1.10.2 Desktop

Native

platform = native

Native development platform is intended to be used for desktop OS. This platform uses built-in toolchains (preferable based on GCC), frameworks, libs from particular OS where it will be run.

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Stable and upstream versions*

Examples

Examples are listed from Native development platform repository:

- hello-world

Stable and upstream versions

You can switch between stable releases of Native development platform and the latest upstream version using *platform* option in *Project Configuration File* `platformio.ini` as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = native
board = ...

; Custom stable version
[env:custom_stable]
platform = native@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-native.git
board = ...
```

Linux ARM

platform = linux_arm

Linux ARM is a Unix-like and mostly POSIX-compliant computer operating system (OS) assembled under the model of free and open-source software development and distribution. Using host OS (Mac OS X, Linux ARM) you can build native application for Linux ARM platform.

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Stable and upstream versions*
- *Packages*
- *Frameworks*
- *Boards*

Examples

Examples are listed from [Linux ARM development platform repository](#):

- wiringpi-blink
- wiringpi-serial

Stable and upstream versions

You can switch between stable releases of Linux ARM development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = linux_arm
board = ...

; Custom stable version
[env:custom_stable]
platform = linux_arm@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-linux_arm.git
board = ...
```

Packages

Name	Description
framework-artik-sdk	ARTIK SDK is a C/C++ SDK targeting Samsung ARTIK platforms
framework-wiringpi	GPIO Interface library for the Raspberry Pi
toolchain-gccarmlinuxgnueabi	GCC for Linux ARM GNU EABI

Frameworks

Name	Description
AR-TIK SDK	ARTIK SDK is a C/C++ SDK targeting Samsung ARTIK platforms. It exposes a set of APIs to ease up development of applications. These APIs cover hardware buses such as GPIO, SPI, I2C, UART, connectivity links like Wi-Fi, Bluetooth, Zigbee, and network protocols such as HTTP, Websockets, MQTT, and others.
Wiring	WiringPi is a GPIO access library written in C for the BCM2835 used in the Raspberry Pi. It's designed to be familiar to people who have used the Arduino "wiring" system.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

Raspberry Pi

ID	Name	Debug	MCU	Frequency	Flash	RAM
raspberrypi_1b	Raspberry Pi 1 Model B	No	BCM2835	700MHz	512MB	512MB
raspberrypi_2b	Raspberry Pi 2 Model B	No	BCM2836	900MHz	1GB	1GB
raspberrypi_3b	Raspberry Pi 3 Model B	No	BCM2837	1200MHz	1GB	1GB
raspberrypi_zero	Raspberry Pi Zero	No	BCM2835	1000MHz	512MB	512MB

RushUp

ID	Name	Debug	MCU	Frequency	Flash	RAM
kitra_520	RushUp Kitra 520	No	EXYNOS3250	1000MHz	4GB	512MB

Samsung

ID	Name	Debug	MCU	Frequency	Flash	RAM
artik_1020	Samsung ARTIK 1020	No	EXYNOS5422	1500MHz	16GB	2GB
artik_520	Samsung ARTIK 520	No	EXYNOS3250	1000MHz	4GB	512MB
artik_530	Samsung ARTIK 530	No	S5P4418	1200MHz	4GB	512MB
artik_710	Samsung ARTIK 710	No	S5P6818	1400MHz	4GB	1GB

Linux i686

`platform = linux_i686`

Linux i686 (32-bit) is a Unix-like and mostly POSIX-compliant computer operating system (OS) assembled under the model of free and open-source software development and distribution. Using host OS (Mac OS X or Linux 32-bit) you can build native application for Linux i686 platform.

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Stable and upstream versions*
- *Packages*

Examples

Examples are listed from Linux i686 development platform repository:

- hello-world

Stable and upstream versions

You can switch between stable releases of Linux i686 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = linux_i686
board = ...

; Custom stable version
[env:custom_stable]
platform = linux_i686@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-linux_i686.git
board = ...
```

Packages

Name	Description
toolchain-gcclinux32	GCC for Linux i686

Linux x86_64

```
platform = linux_x86_64
```

Linux x86_64 (64-bit) is a Unix-like and mostly POSIX-compliant computer operating system (OS) assembled under the model of free and open-source software development and distribution. Using host OS (Mac OS X or Linux 64-bit) you can build native application for Linux x86_64 platform.

For more detailed information please visit [vendor site](#).

Contents

- Examples

- *Stable and upstream versions*
- *Packages*

Examples

Examples are listed from Linux x86_64 development platform repository:

- hello-world

Stable and upstream versions

You can switch between stable releases of Linux x86_64 development platform and the latest upstream version using *platform* option in *Project Configuration File platformio.ini* as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = linux_x86_64
board = ...

; Custom stable version
[env:custom_stable]
platform = linux_x86_64@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-linux_x86_64.git
board = ...
```

Packages

Name	Description
toolchain-gcclinux64	GCC for Linux x86_64

Windows x86

platform = windows_x86

Windows x86 (32-bit) is a metafamily of graphical operating systems developed and marketed by Microsoft. Using host OS (Windows, Linux 32/64 or Mac OS X) you can build native application for Windows x86 platform.

For more detailed information please visit [vendor site](#).

Contents

- [Examples](#)
- [Stable and upstream versions](#)
- [Packages](#)

Examples

Examples are listed from Windows x86 development platform repository:

- [hello-world](#)

Stable and upstream versions

You can switch between stable releases of Windows x86 development platform and the latest upstream version using [platform](#) option in [Project Configuration File platformio.ini](#) as described below.

Stable

```
; Latest stable version
[env:latest_stable]
platform = windows_x86
board = ...

; Custom stable version
[env:custom_stable]
platform = windows_x86@x.y.z
board = ...
```

Upstream

```
[env:upstream_develop]
platform = https://github.com/platformio/platform-windows_x86.git
board = ...
```

Packages

Name	Description
toolchain-gccmingw32	MinGW

1.11 Frameworks

1.11.1 Arduino

framework = arduino

Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.

For more detailed information please visit [vendor site](#).

Contents

- [*Tutorials*](#)
- [*Debugging*](#)
- [*Examples*](#)
- [*Platforms*](#)
- [*Boards*](#)

Tutorials

- [*Arduino and Nordic nRF52-DK: debugging and unit testing*](#)

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- [*Debug Tools*](#)
 - [*On-Board Debug Tools*](#)
 - [*External Debug Tools*](#)

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Bands listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
bbcmicrobit	BBC micro:bit	Nordic CMSIS-DAP (on-board) nRF51		NRF51	82MHz	256KB	16KB
delta_dongle	Dongle nRF52-BM-NQ620	Nordic CMSIS-DAP (on-board), Black Magic Probe, nRF52 J-LINK, ST-LINK		NRF52	86MHz	512KB	64KB
disco_f401vg	STM32F4DISCO	ST	ST-LINK (default, on-board), Black Magic Probe, J-LINK	STM32F401VG	88MHz	2MB	128KB
esp-wroverkit	Espressif ESP-WROVER-KIT	Espressif FTDI Chip (default, on-board), Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY		ESP32	240MHz	4MB	320KB
mxchip_az3166ft	Azure IoT Development Kit (MXChip AZ3166)	ST	ST-LINK (default, on-board), Black Magic Probe, J-LINK	STM32F401VG	102MHz	2MB	256KB
mzeropro	Arduino M0 Pro (Programming/Debug Port)	Atmel SAM	CMSIS-DAP (on-board), Atmel-ICE, Black Magic Probe, J-LINK	SAMD21G18A	48MHz	256KB	32KB
nrf51_dk	Nordic nRF51-DK	Nordic CMSIS-DAP (on-board), J-LINK (on-board), nRF51 Black Magic Probe, ST-LINK		NRF51	82MHz	256KB	32KB
nrf51_dongle	Nordic nRF51-Dongle	Nordic CMSIS-DAP (on-board), J-LINK (on-board), nRF51		NRF51	82MHz	256KB	32KB
nrf52840_nrf52840-dk	Nordic nRF52840-DK	Nordic CMSIS-DAP (on-board), J-LINK (on-board), nRF52 Black Magic Probe, ST-LINK		NRF52	86MHz	1MB	256KB
nrf52_dk	Nordic nRF52-DK	Nordic CMSIS-DAP (on-board), J-LINK (on-board), nRF52 Black Magic Probe, ST-LINK		NRF52	86MHz	512KB	64KB
nucleo_f103rb	Nucleo F103RB	ST	ST-LINK (default, on-board), Black Magic Probe, J-LINK	STM32F103RBT6	8MHz	28KB	20KB
redBearLab_nRF51822	RedBearLab nRF51822	Nordic CMSIS-DAP (on-board), Black Magic Probe, nRF51 J-LINK, ST-LINK		NRF51	82MHz	256KB	16KB
redBearLab_BLE_Nano_1.5	RedBearLab BLE Nano 1.5	Nordic CMSIS-DAP (on-board), Black Magic Probe, nRF51 J-LINK, ST-LINK		NRF51	82MHz	256KB	32KB
redbear_ble_nano_2	RedBearLab BLE Nano 2	Nordic CMSIS-DAP (on-board), Black Magic Probe, nRF52 J-LINK, ST-LINK		NRF52	86MHz	512KB	64KB
redbear_ble_nano_2_blend_2	RedBearLab Blend 2	Nordic CMSIS-DAP (on-board), Black Magic Probe, nRF52 J-LINK, ST-LINK		NRF52	86MHz	512KB	64KB
samd21g18anel	Atmel ATSAMW25-XPRO	Atmel SAM	CMSIS-DAP (on-board), Atmel-ICE, Black Magic Probe, J-LINK	SAMD21G18A	48MHz	256KB	32KB
seeedtinytina	TinyTina BLE	Nordic CMSIS-DAP (on-board), Black Magic Probe, nRF51 J-LINK, ST-LINK		NRF51	82MHz	256KB	16KB
ublox_evk_nina_fk-nina-b1	EVK-NINA-B1	Nordic J-LINK (on-board), Black Magic Probe, ST-nRF52 LINK		NRF52	86MHz	512KB	64KB
xmc1100_xmc1100-bootkit	XMC1100 Boot Kit	In-fi-neon XMC	J-LINK (on-board)	XMC1100	102MHz	64KB	64KB
xmc1100_xmc1100e2go	XMC1100e2go H-Bridge 2Go	In-fi-neon XMC	J-LINK (on-board)	XMC1100	102MHz	64KB	64KB
342_xmc1100_xmc2go	XMC1100 XMC2Go	In-fi-neon XMC	J-LINK (on-board)	XMC1100	102MHz	64KB	64KB

External Debug Tools

Boards listed below are compatible with [PIO Unified Debugger](#) but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Platform	Debug
adafruit_circuitplayground_m0	Adafruit Circuit Playground Express	Atmel SAM	Atmel-ICE, Black Magic
adafruit_feather_m0	Adafruit Feather M0	Atmel SAM	Atmel-ICE, Black Magic
adafruit_feather_m0_express	Adafruit Feather M0 Express	Atmel SAM	Atmel-ICE, Black Magic
adafruit_feather_m4	Adafruit Feather M4 (SAMD51)	Atmel SAM	Atmel-ICE, Black Magic
adafruit_gemma_m0	Adafruit Gemma M0	Atmel SAM	Atmel-ICE, Black Magic
adafruit_itsybitsy_m0	Adafruit ItsyBitsy M0	Atmel SAM	Atmel-ICE, Black Magic
adafruit_itsybitsy_m4	Adafruit ItsyBitsy M4 (SAMD51)	Atmel SAM	Atmel-ICE, Black Magic
adafruit.metro.m0	Adafruit Metro M0 Express	Atmel SAM	Atmel-ICE, Black Magic
adafruit.metro.m4	Adafruit Metro M4 (SAMD51)	Atmel SAM	Atmel-ICE, Black Magic
adafruit_pirkey	Adafruit pIRkey	Atmel SAM	Atmel-ICE, Black Magic
adafruit_trinket_m0	Adafruit Trinket M0	Atmel SAM	Atmel-ICE, Black Magic
alksesp32	ALKS ESP32	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
bluepill_f103c8	BluePill F103C8	ST STM32	Black Magic Probe, J-LI
bluz_dk	BluzDK	Nordic nRF51	Black Magic Probe, J-LI
digix	Digistump DigiX	Atmel SAM	Black Magic Probe, J-LI
due	Arduino Due (Programming Port)	Atmel SAM	Black Magic Probe, J-LI
dueUSB	Arduino Due (USB Native Port)	Atmel SAM	Black Magic Probe, J-LI
esp32-evb	OLIMEX ESP32-EVB	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
esp32-gateway	OLIMEX ESP32-GATEWAY	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
esp32dev	Espressif ESP32 Dev Module	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
esp32doit-devkit-v1	DOIT ESP32 DEVKIT V1	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
esp32thing	SparkFun ESP32 Thing	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
esp32vn-iot-uno	ESP32vn IoT Uno	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
espectro32	ESPectro32	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
espino32	ESPino32	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
featheresp32	Adafruit ESP32 Feather	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
firebeetle32	FireBeetle-ESP32	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
genericSTM32F103C8	STM32F103C8 (20k RAM. 64k Flash)	ST STM32	Black Magic Probe, J-LI
genericSTM32F103CB	STM32F103CB (20k RAM. 128k Flash)	ST STM32	Black Magic Probe, J-LI
genericSTM32F103R8	STM32F103R8 (20k RAM. 64 Flash)	ST STM32	Black Magic Probe, J-LI
genericSTM32F103RB	STM32F103RB (20k RAM. 128k Flash)	ST STM32	Black Magic Probe, J-LI
genericSTM32F103RC	STM32F103RC (48k RAM. 256k Flash)	ST STM32	Black Magic Probe, J-LI
genericSTM32F103RE	STM32F103RE (64k RAM. 512k Flash)	ST STM32	Black Magic Probe, J-LI
genericSTM32F103VC	STM32F103VC (48k RAM. 256k Flash)	ST STM32	Black Magic Probe, J-LI
genericSTM32F103VE	STM32F103VE (64k RAM. 512k Flash)	ST STM32	Black Magic Probe, J-LI
genericSTM32F407VET6	STM32F407VE (192k RAM. 512k Flash)	ST STM32	ST-LINK
hornbill32dev	Hornbill ESP32 Dev	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
hornbill32minima	Hornbill ESP32 Minima	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
lolin32	WEMOS LOLIN32	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
lolin_d32	WEMOS LOLIN D32	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
lolin_d32_pro	WEMOS LOLIN D32 PRO	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LI
macchina2	Macchina M2	Atmel SAM	Black Magic Probe, J-LI
maple	Maple	ST STM32	Black Magic Probe, J-LI
maple_mini_b20	Maple Mini Bootloader 2.0	ST STM32	Black Magic Probe, J-LI
maple_mini_origin	Maple Mini Original	ST STM32	Black Magic Probe, J-LI

ID	Name	Platform	Debug
mhetesp32devkit	MH ET LIVE ESP32DevKIT	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LINK
mhetesp32minikit	MH ET LIVE ESP32MiniKit	Espressif 32	Mini-Module FT2232H, Atmel-ICE, Black Magic Probe, J-LINK
mkrl000USB	Arduino MKR1000	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
mkrfox1200	Arduino MKR FOX 1200	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
mkrgsm1400	Arduino MKR GSM 1400	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
mkewan1300	Arduino MKR WAN 1300	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
mkrzero	Arduino MKRZERO	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
mzeroUSB	Arduino M0	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
mzeroprousb	Arduino M0 Pro (Native USB Port)	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
ng_beacon	ng-beacon	Nordic nRF51	Black Magic Probe, J-LINK
node32s	Node32s	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LINK
nodemcu-32s	NodeMCU-32S	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LINK
oshchip	OSHChip	Nordic nRF51	Black Magic Probe, J-LINK
pocket_32	Dongsen Tech Pocket 32	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LINK
rfduino	RFduino	Nordic nRF51	Black Magic Probe, J-LINK
sainSmartDue	SainSmart Due (Programming Port)	Atmel SAM	Black Magic Probe, J-LINK
sainSmartDueUSB	SainSmart Due (USB Native Port)	Atmel SAM	Black Magic Probe, J-LINK
sodaq_autonomo	SODAQ Autonomo	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
sodaq_explorer	SODAQ ExpLoRer	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
sodaq_one	SODAQ ONE	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
sparkfun_samd21_dev_usb	SparkFun SAMD21 Dev Breakout	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
sparkfun_samd21_mini_usb	SparkFun SAMD21 Mini Breakout	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
stct_nrf52_minidev	Taida Century nRF52 mini board	Nordic nRF52	Black Magic Probe, J-LINK
teensy31	Teensy 3.1 / 3.2	Teensy	J-LINK
teensy35	Teensy 3.5	Teensy	J-LINK
teensy36	Teensy 3.6	Teensy	J-LINK
teensylc	Teensy LC	Teensy	J-LINK
tian	Arduino Tian	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK
ttgo-lora32-v1	TTGO LoRa32-OLED V1	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LINK
waveshare_ble400	Waveshare BLE400	Nordic nRF51	Black Magic Probe, J-LINK
wemosbat	WeMos WiFi & Bluetooth Battery	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LINK
xinabox_cw02	XinaBox CW02	Espressif 32	Mini-Module FT2232H, Black Magic Probe, J-LINK
zeroUSB	Arduino Zero (USB Native Port)	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK

Examples

- Arduino for Atmel AVR
- Arduino for Atmel SAM
- Arduino for Espressif 32
- Arduino for Espressif 8266
- Arduino for Infineon XMC
- Arduino for Intel ARC32
- Arduino for Microchip PIC32
- Arduino for Nordic nRF51
- Arduino for Nordic nRF52

- Arduino for ST STM32
- Arduino for Teensy
- Arduino for TI MSP430

Platforms

Name	Description
Atmel AVR	Atmel AVR 8- and 32-bit MCUs deliver a unique combination of performance, power efficiency and design flexibility. Optimized to speed time to market-and easily adapt to new ones-they are based on the industry's most code-efficient architecture for C and assembly programming.
Atmel SAM	Atmel SMART offers Flash- based ARM products based on the ARM Cortex-M0+, Cortex-M3 and Cortex-M4 architectures, ranging from 8KB to 2MB of Flash including a rich peripheral and feature mix.
Espressif 32	Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.
Espressif 8266	Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.
Infineon XMC	Infineon has designed the XMC microcontrollers for real-time critical applications with an industry-standard core. The XMC microcontrollers can be integrated with the Arduino platform
Intel ARC32	ARC embedded processors are a family of 32-bit CPUs that are widely used in SoC devices for storage, home, mobile, automotive, and Internet of Things applications.
Microchip PIC32	Microchip's 32-bit portfolio with the MIPS microAptiv or M4K core offer high performance microcontrollers, and all the tools needed to develop your embedded projects. PIC32 MCUs gives your application the processing power, memory and peripherals your design needs!
Nordic nRF51	The Nordic nRF51 Series is a family of highly flexible, multi-protocol, system-on-chip (SoC) devices for ultra-low power wireless applications. nRF51 Series devices support a range of protocol stacks including Bluetooth Smart (previously called Bluetooth low energy), ANT and proprietary 2.4GHz protocols such as Gazell.
Nordic nRF52	The nRF52 Series are built for speed to carry out increasingly complex tasks in the shortest possible time and return to sleep, conserving precious battery power. They have a Cortex-M4F processor and are the most capable Bluetooth Smart SoCs on the market.
ST STM32	The STM32 family of 32-bit Flash MCUs based on the ARM Cortex-M processor is designed to offer new degrees of freedom to MCU users. It offers a 32-bit product range that combines very high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.
Teensy	Teensy is a complete USB-based microcontroller development system, in a very small footprint, capable of implementing many types of projects. All programming is done via the USB port. No special programmer is needed, only a standard USB cable and a PC or Macintosh with a USB port.
TI MSP430	MSP430 microcontrollers (MCUs) from Texas Instruments (TI) are 16-bit, RISC-based, mixed-signal processors designed for ultra-low power. These MCUs offer the lowest power consumption and the perfect mix of integrated peripherals for thousands of applications.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer

- For more detailed board information please scroll tables below by horizontal.
-

4D Systems

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
gen4iod	4D Systems gen4 IoD Range	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB

4DSsystems

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
picadillo_35t	4DSsystems PI-Cadillo 35T	<i>Microchip PIC32</i>	No	32MX795F512L	80MHz	508KB	128KB

Adafruit

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
adafruit_circuitplay	Adafruit Circuit Playground Express	<i>Atmel SAM</i>	Yes	SAMD21G188MHz	256KB	32KB	
adafruit_feather_m0	Adafruit Feather M0	<i>Atmel SAM</i>	Yes	SAMD21G188MHz	256KB	32KB	
adafruit_feather_m0	Adafruit Feather M0 Express	<i>Atmel SAM</i>	Yes	SAMD21G188MHz	256KB	32KB	
adafruit_feather_m4	Adafruit Feather M4 (SAMD51)	<i>Atmel SAM</i>	Yes	SAMD51J19120MHz	496KB	192KB	
adafruit_gemma_m0	Adafruit Gemma M0	<i>Atmel SAM</i>	Yes	SAMD21E188MHz	256KB	32KB	
adafruit_itsybitsy_m0	Adafruit ItsyBitsy M0	<i>Atmel SAM</i>	Yes	SAMD21G188MHz	256KB	32KB	
adafruit_itsybitsy_m0	Adafruit ItsyBitsy M4 (SAMD51)	<i>Atmel SAM</i>	Yes	SAMD51J19120MHz	496KB	192KB	
adafruit.metro_m0	Adafruit Metro M0 Express	<i>Atmel SAM</i>	Yes	SAMD21G188MHz	256KB	32KB	
adafruit.metro_m4	Adafruit Metro M4 (SAMD51)	<i>Atmel SAM</i>	Yes	SAMD51J19120MHz	496KB	192KB	
adafruit_pirkey	Adafruit pIRkey	<i>Atmel SAM</i>	Yes	SAMD21E188MHz	256KB	32KB	
adafruit_trinket_m0	Adafruit Trinket M0	<i>Atmel SAM</i>	Yes	SAMD21E188MHz	256KB	32KB	
bluefruitmicro	Adafruit Bluefruit Micro	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
feather328p	Adafruit Feather 328P	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
feather32u4	Adafruit Feather 32u4	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
featheresp32	Adafruit ESP32 Feather	<i>Espres-sif 32</i>	Yes	ESP32	240MHz	4MB	320KB
flora8	Adafruit Flora	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
gemma	Adafruit Gemma	<i>Atmel AVR</i>	No	AT-TINY85	8MHz	8KB	512B
huzzah	Adafruit HUZZAH	<i>Espres-sif 8266</i>	No	ESP8266	80MHz	4MB	80KB
itsybitsy32u4_3V	Adafruit ItsyBitsy 3V/8MHz	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
itsybitsy32u4_5V	Adafruit ItsyBitsy 5V/16MHz	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
metro	Adafruit Metro	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB
protrinket3	Adafruit Pro Trinket 3V/12MHz (USB)	<i>Atmel AVR</i>	No	AT-MEGA328P	12MHz	28KB	2KB
protrinket3ftdi	Adafruit Pro Trinket 3V/12MHz (FTDI)	<i>Atmel AVR</i>	No	AT-MEGA328P	12MHz	28KB	2KB
protrinket5	Adafruit Pro Trinket 5V/16MHz (USB)	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	28KB	2KB
protrinket5ftdi	Adafruit Pro Trinket 5V/16MHz (FTDI)	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	28KB	2KB
trinket3	Adafruit Trinket 3V/8MHz	<i>Atmel AVR</i>	No	AT-TINY85	8MHz	8KB	1KB
trinket5	Adafruit Trinket 5V/16MHz	<i>Atmel AVR</i>	No	AT-TINY85	16MHz	8KB	512B

Aiyarafun

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
node32s	Node32s	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

Alorium Technology

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
alorium_xlr8	Alorium XLR8	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB

Anarduino

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
miniwireless	Anarduino MiniWireless	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB

April Brother

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espea32	April Brother ESPeA32	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

Arduboy

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
arduboy	Arduboy	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
arduboy_devkit	Arduboy DevKit	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB

Arduino

ID	Name	Platform	Debug	MCU
LilyPadUSB	Arduino LilyPad USB	<i>Atmel AVR</i>	No	ATMEGA32U4
atmega328pb	Atmel ATmega328PB	<i>Atmel AVR</i>	No	ATMEGA328PB
atmegangatmega168	Arduino NG or older ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
atmegangatmega8	Arduino NG or older ATmega8	<i>Atmel AVR</i>	No	ATMEGA8
btatmega168	Arduino BT ATmega168	<i>Atmel AVR</i>	No	ATMEGA168

Table 6 – continued from previous page

ID	Name	Platform	Debug	MCU
btatmega328	Arduino BT ATmega328	<i>Atmel AVR</i>	No	ATMEGA328P
chiwawa	Arduino Industrial 101	<i>Atmel AVR</i>	No	ATMEGA32U4
diecimilaatmega168	Arduino Duemilanove or Diecimila ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
diecimilaatmega328	Arduino Duemilanove or Diecimila ATmega328	<i>Atmel AVR</i>	No	ATMEGA328P
due	Arduino Due (Programming Port)	<i>Atmel SAM</i>	Yes	AT91SAM3X8E
dueUSB	Arduino Due (USB Native Port)	<i>Atmel SAM</i>	Yes	AT91SAM3X8E
esplora	Arduino Esplora	<i>Atmel AVR</i>	No	ATMEGA32U4
ethernet	Arduino Ethernet	<i>Atmel AVR</i>	No	ATMEGA328P
fio	Arduino Fio	<i>Atmel AVR</i>	No	ATMEGA328P
leonardo	Arduino Leonardo	<i>Atmel AVR</i>	No	ATMEGA32U4
leonardoeth	Arduino Leonardo ETH	<i>Atmel AVR</i>	No	ATMEGA32U4
lilypadatmega168	Arduino LilyPad ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
lilypadatmega328	Arduino LilyPad ATmega328	<i>Atmel AVR</i>	No	ATMEGA328P
megaADK	Arduino Mega ADK	<i>Atmel AVR</i>	No	ATMEGA2560
megaatmega1280	Arduino Mega or Mega 2560 ATmega1280	<i>Atmel AVR</i>	No	ATMEGA1280
megaatmega2560	Arduino Mega or Mega 2560 ATmega2560 (Mega 2560)	<i>Atmel AVR</i>	No	ATMEGA2560
micro	Arduino Micro	<i>Atmel AVR</i>	No	ATMEGA32U4
miniatmega168	Arduino Mini ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
miniatmega328	Arduino Mini ATmega328	<i>Atmel AVR</i>	No	ATMEGA328P
mkr1000USB	Arduino MKR1000	<i>Atmel SAM</i>	Yes	SAMD21G18A
mkrfox1200	Arduino MKR FOX 1200	<i>Atmel SAM</i>	Yes	SAMD21G18A
mkrgsm1400	Arduino MKR GSM 1400	<i>Atmel SAM</i>	Yes	SAMD21G18A
mkewan1300	Arduino MKR WAN 1300	<i>Atmel SAM</i>	Yes	SAMD21G18A
mkrzero	Arduino MKRZERO	<i>Atmel SAM</i>	Yes	SAMD21G18A
mzeroUSB	Arduino M0	<i>Atmel SAM</i>	Yes	SAMD21G18A
mzeropro	Arduino M0 Pro (Programming/Debug Port)	<i>Atmel SAM</i>	Yes	SAMD21G18A
mzeroprousb	Arduino M0 Pro (Native USB Port)	<i>Atmel SAM</i>	Yes	SAMD21G18A
nanoatmega168	Arduino Nano ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
nanoatmega328	Arduino Nano ATmega328	<i>Atmel AVR</i>	No	ATMEGA328P
pro16MHzatmega168	Arduino Pro or Pro Mini ATmega168 (5V, 16 MHz)	<i>Atmel AVR</i>	No	ATMEGA168
pro16MHzatmega328	Arduino Pro or Pro Mini ATmega328 (5V, 16 MHz)	<i>Atmel AVR</i>	No	ATMEGA328P
pro8MHzatmega168	Arduino Pro or Pro Mini ATmega168 (3.3V, 8 MHz)	<i>Atmel AVR</i>	No	ATMEGA168
pro8MHzatmega328	Arduino Pro or Pro Mini ATmega328 (3.3V, 8 MHz)	<i>Atmel AVR</i>	No	ATMEGA328P
robotControl	Arduino Robot Control	<i>Atmel AVR</i>	No	ATMEGA32U4
robotMotor	Arduino Robot Motor	<i>Atmel AVR</i>	No	ATMEGA32U4
tian	Arduino Tian	<i>Atmel SAM</i>	Yes	SAMD21G18A
uno	Arduino Uno	<i>Atmel AVR</i>	No	ATMEGA328P
yun	Arduino Yun	<i>Atmel AVR</i>	No	ATMEGA32U4
yunmini	Arduino Yun Mini	<i>Atmel AVR</i>	No	ATMEGA32U4
zero	Arduino Zero (Programming/Debug Port)	<i>Atmel SAM</i>	Yes	SAMD21G18A
zeroUSB	Arduino Zero (USB Native Port)	<i>Atmel SAM</i>	Yes	SAMD21G18A

Atmel

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
attiny13	Generic ATTiny13	<i>Atmel AVR</i>	No	ATTINY13	9MHz	1KB	64B
attiny1634	Generic ATTiny1634	<i>Atmel AVR</i>	No	ATTINY1634	8MHz	16KB	1KB
attiny167	Generic ATTiny167	<i>Atmel AVR</i>	No	ATTINY167	8MHz	16KB	512B
attiny2313	Generic ATTiny2313	<i>Atmel AVR</i>	No	ATTINY2313	8MHz	2KB	128B
attiny24	Generic ATTiny24	<i>Atmel AVR</i>	No	ATTINY24	8MHz	2KB	128B
attiny25	Generic ATTiny25	<i>Atmel AVR</i>	No	ATTINY25	8MHz	2KB	128B
attiny261	Generic ATTiny261	<i>Atmel AVR</i>	No	ATTINY261	8MHz	2KB	128B
attiny4313	Generic ATTiny4313	<i>Atmel AVR</i>	No	ATTINY4313	8MHz	4KB	256B
attiny44	Generic ATTiny44	<i>Atmel AVR</i>	No	ATTINY44	8MHz	4KB	256B
attiny441	Generic ATTiny441	<i>Atmel AVR</i>	No	ATTINY441	8MHz	4KB	256B
attiny45	Generic ATTiny45	<i>Atmel AVR</i>	No	ATTINY45	8MHz	4KB	256B
attiny461	Generic ATTiny461	<i>Atmel AVR</i>	No	ATTINY461	8MHz	4KB	256B
attiny48	Generic ATTiny48	<i>Atmel AVR</i>	No	ATTINY48	8MHz	4KB	256B
attiny828	Generic ATTiny828	<i>Atmel AVR</i>	No	ATTINY828	8MHz	8KB	512B
attiny84	Generic ATTiny84	<i>Atmel AVR</i>	No	ATTINY84	8MHz	8KB	512B
attiny841	Generic ATTiny841	<i>Atmel AVR</i>	No	ATTINY841	8MHz	8KB	512B
attiny85	Generic ATTiny85	<i>Atmel AVR</i>	No	ATTINY85	8MHz	8KB	512B
attiny861	Generic ATTiny861	<i>Atmel AVR</i>	No	ATTINY861	8MHz	8KB	512B
attiny87	Generic ATTiny87	<i>Atmel AVR</i>	No	ATTINY87	8MHz	8KB	512B
attiny88	Generic ATTiny88	<i>Atmel AVR</i>	No	ATTINY88	8MHz	8KB	512B
samd21g18a	Atmel ATSAMW25-XPRO	<i>Atmel SAM</i>	Yes	SAMD21G18A	48MHz	256KB	32KB
usbasp	USBasp stick	<i>Atmel AVR</i>	No	ATMEGA8	12MHz	8KB	1KB

BBC

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
bbcmicrobit	BBC micro:bit	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	256KB	16KB

BQ

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
zumbt328	BQ ZUM BT-328	<i>Atmel AVR</i>	No	ATMEGA328P	16MHz	28KB	2KB

BitWizard

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
raspduino	BitWizard Raspduino	<i>Atmel AVR</i>	No	ATMEGA328P	16MHz	30KB	2KB

BluzDK

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
bluz_dk	BluzDK	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB

Controllino

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
controllino_maxi	Controllino Maxi	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	248KB	8KB
controllino_maxi_automation	Controllino Maxi Automation	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	248KB	8KB
controllino_mega	Controllino Mega	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	248KB	8KB
controllino_mini	Controllino Mini	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB

DFRobot

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
firebeetle32	FireBeetle-ESP32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

DOIT

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
esp32doit-devkit-v1	DOIT ESP32 DEVKIT V1	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Delta

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
delta_dfbm_nq620	Delta DFBM-NQ620	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

DigiStump

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
oak	DigiStump Oak	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Digilent

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
cerebot32mx4	Digilent Cerebot 32MX4	Microchip PIC32	No	32MX460F512L	80MHz	508KB	32KB
cerebot32mx7	Digilent Cerebot 32MX7	Microchip PIC32	No	32MX795F512L	80MHz	508KB	128KB
chipkit_cmod	Digilent chipKIT Cmod	Microchip PIC32	No	32MX150F128D	40MHz	124KB	32KB
chipkit_dp32	Digilent chipKIT DP32	Microchip PIC32	No	32MX250F128B	40MHz	120KB	32KB
chipkit_mx3	Digilent chipKIT MX3	Microchip PIC32	No	32MX320F128H	80MHz	124KB	16KB
chipkit_pro_mx4	Digilent chipKIT Pro MX4	Microchip PIC32	No	32MX460F512L	80MHz	508KB	32KB
chipkit_pro_mx7	Digilent chipKIT Pro MX7	Microchip PIC32	No	32MX795F512L	80MHz	508KB	128KB
chipkit_uc32	Digilent chipKIT uC32	Microchip PIC32	No	32MX340F512H	80MHz	508KB	32KB
chipkit_wf32	Digilent chipKIT WF32	Microchip PIC32	No	32MX695F512L	80MHz	508KB	128KB
chipkit_wifire	Digilent chipKIT WiFire	Microchip PIC32	No	32MZ2048ECG1	100MHz	1.98MB	512KB
mega_pic32	Digilent chipKIT MAX32	Microchip PIC32	No	32MX795F512L	80MHz	508KB	128KB
openscope	Digilent OpenScope	Microchip PIC32	No	32MZ2048EFG1	120MHz	1.98MB	512KB
uno_pic32	Digilent chipKIT UNO32	Microchip PIC32	No	32MX320F128H	80MHz	124KB	16KB

Digistump

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
digispark-pro	Digispark Pro	Atmel AVR	No	AT-TINY167	16MHz	14.50KB	512B
digispark-pro	Digispark Pro (32 byte buffer)	Atmel AVR	No	AT-TINY167	16MHz	14.50KB	512B
digispark-pro	Digispark Pro (16 MHz) (64 byte buffer)	Atmel AVR	No	AT-TINY167	16MHz	14.50KB	512B
digispark-tin	Digispark USB	Atmel AVR	No	ATTINY85	16MHz	5.87KB	512B
digix	Digistump DigiX	Atmel SAM	Yes	AT91SAM3X8E	4MHz	512KB	28KB

Doit

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
espduino	ESPDuino (ESP-13 Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Dongsen Technology

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
pocket_32	Dongsen Tech Pocket 32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Dwengo

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
dwenguino	Dwenguino	<i>Atmel AVR</i>	No	AT90USB646	16MHz	60KB	2KB

DycodeX

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espectro	ESPectro Core	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
espectro32	ESPectro32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

ESP32vn

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp32vn-iot-uno	ESP32vn IoT Uno	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

ESPerf

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
espresso_lite_v1	ESPRESSO Lite 1.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
espresso_lite_v2	ESPRESSO Lite 2.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

ESPino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espino	ESPino	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Electronic SweetPeas

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp320	Electronic SweetPeas ESP320	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

Elektor

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
elektor_uno_r4	Elektor Uno R4	<i>Atmel AVR</i>	No	AT-MEGA328PB	16MHz	31.50KB	2KB

Engduino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
engduinov3	Engduino 3	<i>Atmel AVR</i>	No	ATMEGA32U4	8MHz	28KB	2.50KB

EnviroDIY

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mayfly	EnviroDIY Mayfly	<i>Atmel AVR</i>	No	ATMEGA1284P	8MHz	127KB	16KB

Espressif

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
esp-wrover-kit	Espressif ESP-WROVER-KIT	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB
esp01	Espressif Generic ESP8266 ESP-01 512k	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB
esp01_1m	Espressif Generic ESP8266 ESP-01 1M	<i>Espressif 8266</i>	No	ESP8266	80MHz	1MB	80KB
esp07	Espressif Generic ESP8266 ESP-07	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
esp12e	Espressif ESP8266 ESP-12E	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
esp32dev	Espressif ESP32 Dev Module	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB
esp8285	Generic ESP8285 Module	<i>Espressif 8266</i>	No	ESP8266	80MHz	423.98KB	80KB
esp_wroom_02	ESP-WROOM-02	<i>Espressif 8266</i>	No	ESP8266	80MHz	2MB	80KB
phoenix_v1	Phoenix 1.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
phoenix_v2	Phoenix 2.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
pico32	ESP32 Pico Kit	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB
wifinfo	WifiInfo	<i>Espressif 8266</i>	No	ESP8266	80MHz	1MB	80KB

Fubarino

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
fubarino_mini	Fubarino Mini	<i>Microchip PIC32</i>	No	32MX250F128D	48MHz	120KB	32KB
fubarino_sd	Fubarino SD (1.5)	<i>Microchip PIC32</i>	No	32MX795F512H	80MHz	508KB	128KB

Generic

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
bluepill_f103c8	BluePill F103C8	ST STM32	Yes	STM32F103C8T6MHz	8MHz	64KB	20KB
genericSTM32F103C8	STM32F103C8 (20k RAM. 64k Flash)	ST STM32	Yes	STM32F103C8T6MHz	8MHz	64KB	20KB
genericSTM32F103CB	STM32F103CB (20k RAM. 128k Flash)	ST STM32	Yes	STM32F103CBT6MHz	128MHz	128KB	20KB
genericSTM32F103R8	STM32F103R8 (20k RAM. 64 Flash)	ST STM32	Yes	STM32F103RT6MHz	8MHz	64KB	20KB
genericSTM32F103RB	STM32F103RB (20k RAM. 128k Flash)	ST STM32	Yes	STM32F103RBT6MHz	128MHz	128KB	20KB
genericSTM32F103RC	STM32F103RC (48k RAM. 256k Flash)	ST STM32	Yes	STM32F103RCT6MHz	128MHz	256KB	48KB
genericSTM32F103RE	STM32F103RE (64k RAM. 512k Flash)	ST STM32	Yes	STM32F103RET6MHz	128MHz	512KB	64KB
genericSTM32F103VC	STM32F103VC (48k RAM. 256k Flash)	ST STM32	Yes	STM32F103VCT6MHz	128MHz	256KB	48KB
genericSTM32F103VE	STM32F103VE (64k RAM. 512k Flash)	ST STM32	Yes	STM32F103VET6MHz	128MHz	512KB	64KB
genericSTM32F407VE	STM32F407VE (192k RAM. 512k Flash)	ST STM32	Yes	STM32F407VET6MHz	128MHz	502.23KB	128KB

Hardkernel

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
odroid_esp32	ODROID-GO	Espressif 32	No	ESP32	240MHz	16MB	320KB

Heltec

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
heltec_wifi_kit_8	Heltec Wifi kit 8	Espressif 8266	No	ESP8266	80MHz	4MB	80KB

Heltec Automation

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
heltec_wifi_kit_32	Heltec WIFI Kit 32	Espressif 32	No	ESP32	240MHz	4MB	320KB
heltec_wifi_lora_32	Heltec WIFI LoRa 32	Espressif 32	No	ESP32	240MHz	4MB	320KB

Hornbill

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
hornbill132dev	Hornbill ESP32 Dev	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
hornbill132minima	Hornbill ESP32 Minima	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Infineon

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
xmc1100_boot_kit	XMC1100 Boot Kit	<i>Infineon XMC</i>	<i>Yes</i>	XMC1100	32MHz	64KB	64KB
xmc1100_h_bridge2go	XMC1100 H-Bridge 2Go	<i>Infineon XMC</i>	<i>Yes</i>	XMC1100	32MHz	64KB	64KB
xmc1100_xmc2go	XMC1100 XMC2Go	<i>Infineon XMC</i>	<i>Yes</i>	XMC1100	32MHz	64KB	64KB
xmc1300_boot_kit	XMC1300 Boot Kit	<i>Infineon XMC</i>	<i>Yes</i>	XMC1300	32MHz	64KB	64KB
xmc1300_sense2go	XMC1300 Sense2GoL	<i>Infineon XMC</i>	<i>Yes</i>	XMC1300	32MHz	64KB	122.23KB
xmc4200_distance2go	XMC4200 Distance2Go	<i>Infineon XMC</i>	<i>Yes</i>	XMC4200	80MHz	250KB	256KB
xmc4700_relax_kit	XMC4700 Relax Kit	<i>Infineon XMC</i>	<i>Yes</i>	XMC4700	144MHz	2.00MB	1.95MB

Intel

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
genuino101	Arduino/Genuino 101	<i>Intel ARC32</i>	No	ARCV2EM	32MHz	152KB	80KB

IntoRobot

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
intorobot	IntoRobot Fig	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

LeafLabs

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
maple	Maple	<i>ST STM32</i>	Yes	STM32F103RB	67MHz	108KB	17KB
maple_mini_b20	Maple Mini Boot-loader 2.0	<i>ST STM32</i>	Yes	STM32F103CB	67MHz	120KB	20KB
maple_mini_origin	Maple Mini Original	<i>ST STM32</i>	Yes	STM32F103CB	67MHz	108KB	17KB

LightUp

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lightup	LightUp	<i>Atmel AVR</i>	No	ATMEGA32U4	8MHz	28KB	2.50KB

Linino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
one	Linino One	<i>Atmel AVR</i>	No	ATMEGA32U4	16MHz	28KB	2.50KB

LowPowerLab

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
mightyhat	LowPowerLab MightyHat	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31KB	2KB
moteino	LowPowerLab Moteino	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB
moteinomega	LowPowerLab MoteinoMEGA	<i>Atmel AVR</i>	No	AT-MEGA1284P	16MHz	127KB	16KB

M5Stack

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
m5stack-core-esp32	M5Stack Core ESP32	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB
m5stack-fire	M5Stack FIRE	<i>Espressif 32</i>	No	ESP32	240MHz	16MB	320KB

MH-ET Live

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
mhetesp32devkit	MH ET LIVE ESP32DevKIT	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
mhetesp32minikit	MH ET LIVE ESP32MiniKit	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

MXChip

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
mxchip_az31	Microsoft Azure IoT Development Kit (MXChip AZ3166)	<i>ST STM32</i>	<i>Yes</i>	STM32F412ZG	100MHz	1MB	256KB

Macchina

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
macchina2	Macchina M2	<i>Atmel SAM</i>	<i>Yes</i>	AT91SAM3X8E	84MHz	512KB	32KB

MakerAsia

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
nano32	MakerAsia Nano32	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

Mcudude

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
mightycore1284	MightyCore mega1284	<i>AT- AVR</i>	<i>Atmel AVR</i>	No	AT-MEGA1284P	16MHz	127KB
mightycore16	MightyCore mega16	<i>AT- AVR</i>	<i>Atmel AVR</i>	No	ATMEGA16	16MHz	15.50KB
mightycore164	MightyCore mega164	<i>AT- AVR</i>	<i>Atmel AVR</i>	No	AT-MEGA164P	16MHz	15.50KB
mightycore32	MightyCore mega32	<i>AT- AVR</i>	<i>Atmel AVR</i>	No	ATMEGA32	16MHz	31.50KB
mightycore324	MightyCore mega324	<i>AT- AVR</i>	<i>Atmel AVR</i>	No	AT-MEGA324P	16MHz	31.50KB
mightycore644	MightyCore mega644	<i>AT- AVR</i>	<i>Atmel AVR</i>	No	AT-MEGA644P	16MHz	63KB
mightycore853	MightyCore mega8535	<i>AT- AVR</i>	<i>Atmel AVR</i>	No	ATMEGA16	16MHz	7.50KB

MediaTek Labs

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
smart7688	LinkIt Smart 7688 Duo	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB

Micoduino

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
1284p16m	Micoduino Core+ (AT-mega1284P@16M,5V)	<i>Atmel AVR</i>	No	AT-MEGA1284P	16MHz	127KB	16KB
1284p8m	Micoduino Core+ (AT-mega1284P@8M,3.3V)	<i>Atmel AVR</i>	No	AT-MEGA1284P	8MHz	127KB	16KB
168pa16m	Micoduino Core (At-mega168PA@16M,5V)	<i>Atmel AVR</i>	No	AT-MEGA168P	16MHz	15.50KB	1KB
168pa8m	Micoduino Core (At-mega168PA@8M,3.3V)	<i>Atmel AVR</i>	No	AT-MEGA168P	8MHz	15.50KB	1KB
328p16m	Micoduino Core (At-mega328P@16M,5V)	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB
328p8m	Micoduino Core (At-mega328P@8M,3.3V)	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
32u416m	Micoduino Core USB (AT-mega32U4@16M,5V)	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
644pa16m	Micoduino Core+ (At-mega644PA@16M,5V)	<i>Atmel AVR</i>	No	AT-MEGA644P	16MHz	63KB	4KB
644pa8m	Micoduino Core+ (At-mega644PA@8M,3.3V)	<i>Atmel AVR</i>	No	AT-MEGA644P	8MHz	63KB	4KB
micoduino-core	Micoduino Core ESP32	<i>Espres-sif 32</i>	No	ESP32	240MHz	4MB	320KB

MikroElektronika

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
clicker2	MikroElektronika Clicker 2	<i>Microchip PIC32</i>	No	32MX460F512L	80MHz	508KB	32KB
flipnclickmz	MikroElektronika Flip N Click MZ	<i>Microchip PIC32</i>	No	32MZ2048EFH1052	1052MHz	1.98MB	512KB

NodeMCU

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
nodemcu	NodeMCU 0.9 (ESP-12 Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
nodemcu-32s	NodeMCU-32S	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
nodemcuv2	NodeMCU 1.0 (ESP-12E Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Noduino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
quantum	Noduino Quantum	<i>Espressif 32</i>	No	ESP32	240MHz	16MB	320KB

Nordic

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
nrf51_dk	Nordic nRF51-DK	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB
nrf51_dongle	Nordic nRF51-Dongle	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB
nrf52840_dk	Nordic nRF52840-DK	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52840	64MHz	1MB	256KB
nrf52_dk	Nordic nRF52-DK	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

OLIMEX

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
esp32-evb	OLIMEX ESP32-EVB	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
esp32-gateway	OLIMEX ESP32-GATEWAY	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

OSHChip

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
oshchip	OSHChip	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB

Olimex

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
modwifi	Olimex MOD-WIFI-ESP8266(-DEV)	<i>Espressif 8266</i>	No	ESP8266	80MHz	2MB	80KB
pinguino3	Olimex PIC32-PINGUINO	<i>Microchip PIC32</i>	No	32MX440F256H	80MHz	252KB	32KB

Onehorse

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
onehorse32dev	Onehorse ESP32 Dev Module	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

OpenBCI

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
openbci	OpenBCI 32bit	<i>Microchip PIC32</i>	No	32MX250F128B	40MHz	120KB	32KB

OpenEnergyMonitor

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
emonpi	OpenEnergyMonitor emonPi	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	30KB	2KB

PONTECH

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
usbono_pic32	PONTECH UAV100	<i>Microchip PIC32</i>	No	32MX440F512H	80MHz	508KB	32KB

PanStamp

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
panStampAVR	PanStamp AVR	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
panStampNRG	PanStamp NRG 1.1	<i>TI MSP430</i>	No	CC430F5137	12MHz	31.88KB	4KB

Pinoccio

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
pinoccio	Pinoccio Scout	<i>Atmel AVR</i>	No	ATMEGA256RFR2	16MHz	248KB	32KB

Pololu Corporation

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
a-star32U4	Pololu A-Star 32U4	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB

Pontech

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
nofire	Pontech NoFire	<i>Microchip PIC32</i>	No	32MZ2048EFG100 200MHz	1.98MB	512KB	
quick240_usb	Pontech Quick240	<i>Microchip PIC32</i>	No	32MX795F512L	80MHz	508KB	128KB

Punch Through

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
lightblue-bean	LightBlue Bean	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
lightblue-beanplus	LightBlue Bean+	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB

Quirkbot

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
quirkbot	Quirkbot	<i>Atmel AVR</i>	No	ATMEGA32U4	8MHz	28KB	2.50KB

RFduino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
rfduino	RFduino	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	128KB	8KB

RedBearLab

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
blend	RedBearLab Blend	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
blendmicro16	RedBearLab Blend Micro 3.3V/16MHz (overclock)	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
blendmicro8	RedBearLab Blend Micro 3.3V/8MHz	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
redBearLab	RedBearLab nRF51822	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	256KB	16KB
redBearLabBLE	RedBearLab BLE Nano 1.5	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	256KB	32KB
redbear_blend	RedBearLab BLE Nano 2	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52832	64MHz	512KB	64KB
redbear_blend	RedBearLab Blend 2	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

RepRap

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
reprap_rambo	RepRap RAMBo	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	252KB	8KB

RoboticsBrno

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
alksesp32	ALKS ESP32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

SODAQ

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
sodaq_autonomo	SODAQ Autonomo	<i>Atmel SAM</i>	<i>Yes</i>	SAMD21J18A	48MHz	256KB	32KB
sodaq_explorer	SODAQ Ex-pLoRer	<i>Atmel SAM</i>	<i>Yes</i>	SAMD21J18A	48MHz	256KB	32KB
sodaq_galora	SODAQ GaLoRa	<i>Atmel AVR</i>	No	AT-MEGA1284P	8MHz	127KB	16KB
sodaq_mbili	SODAQ Mbili	<i>Atmel AVR</i>	No	AT-MEGA1284P	8MHz	127KB	16KB
sodaq_moja	SODAQ Moja	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
sodaq_ndogo	SODAQ Ndogo	<i>Atmel AVR</i>	No	AT-MEGA1284P	8MHz	127KB	16KB
sodaq_one	SODAQ ONE	<i>Atmel SAM</i>	<i>Yes</i>	SAMD21G18A	48MHz	256KB	32KB
sodaq_tatu	SODAQ Tatu	<i>Atmel AVR</i>	No	AT-MEGA1284P	8MHz	127KB	16KB

ST

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
disco_f407vg	ST STM32F4DISCOVERY	<i>ST STM32</i>	<i>Yes</i>	STM32F407VGT	6168MHz	1MB	128KB
nucleo_f103rb	ST Nucleo F103RB	<i>ST STM32</i>	<i>Yes</i>	STM32F103RBT	672MHz	128KB	20KB

SainSmart

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
sainSmartDue	SainSmart Due (Program-ming Port)	<i>Atmel SAM</i>	<i>Yes</i>	AT91SAM3X8B	4MHz	512KB	32KB
sainSmartDueU	SainSmart Due (USB Na-tive Port)	<i>Atmel SAM</i>	<i>Yes</i>	AT91SAM3X8B	4MHz	512KB	32KB

Sanguino

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
sanguino_atmega1284p8m	Sanguino ATmega1284p (8MHz)	Atmel AVR	No	AT-MEGA1284P	8MHz	127KB	16KB
sanguino_atmega1284p16m	Sanguino ATmega1284p (16MHz)	Atmel AVR	No	AT-MEGA1284P	16MHz	127KB	16KB
sanguino_atmega644pa16m	Sanguino ATmega644 or ATmega644A (16 MHz)	Atmel AVR	No	AT-MEGA644	16MHz	63KB	4KB
sanguino_atmega644pa8m	Sanguino ATmega644 or ATmega644A (8 MHz)	Atmel AVR	No	AT-MEGA644	8MHz	63KB	4KB
sanguino_atmega644p16m	Sanguino ATmega644P or ATmega644PA (16 MHz)	Atmel AVR	No	AT-MEGA644P	16MHz	63KB	4KB
sanguino_atmega644p8m	Sanguino ATmega644P or ATmega644PA (8 MHz)	Atmel AVR	No	AT-MEGA644P	8MHz	63KB	4KB

SeeedStudio

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
cui32stem	SeeedStudio CUI32stem	Microchip PIC32	No	32MX795F512H80MHz		508KB	128KB
seeedTinyBLE	Seeed Tiny BLE	Nordic nRF51	Yes	NRF51822	16MHz	256KB	16KB
seeeduino	Seeeduino	Atmel AVR	No	AT-MEGA328P	16MHz	31.50KB	2KB
wio_node	Wio Node	Espressif 8266	No	ESP8266	80MHz	4MB	80KB

SparkFun

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
sparkfunBlynk	SparkFun Blynk Board	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
sparkfun_digital sandbox	SparkFun Digital Sandbox	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
sparkfun_fiov3	SparkFun Fio V3 3.3V/8MHz	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
sparkfun_makeymakey	SparkFun Makey Makey	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
sparkfun_megamini	SparkFun Mega Pro Mini 3.3V	<i>Atmel AVR</i>	No	AT-MEGA2560	8MHz	252KB	8KB
sparkfun_megapro16	SparkFun Mega Pro 5V/16MHz	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	248KB	8KB
sparkfun_megapro32	SparkFun Mega Pro 3.3V/8MHz	<i>Atmel AVR</i>	No	AT-MEGA2560	8MHz	252KB	8KB
sparkfun_promicro5v	SparkFun Pro Micro 5V/16MHz	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
sparkfun_promicro3v	SparkFun Pro Micro 3.3V/8MHz	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
sparkfun_qduinomini	SparkFun Qduino Mini	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
sparkfun_redboard	SparkFun RedBoard	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB
sparkfun_samd21_devc	SparkFun SAMD21 Dev Breakout	<i>Atmel SAM</i>	Yes	SAMD21G18A48MHz	256KB	32KB	
sparkfun_samd21_minib	SparkFun SAMD21 Mini Breakout	<i>Atmel SAM</i>	Yes	SAMD21G18A48MHz	256KB	32KB	
sparkfun_satmega128	SparkFun AT-mega128RFA1 Dev Board	<i>Atmel AVR</i>	No	AT-MEGA128RFA1	16MHz	16KB	124KB
sparkfun_serial7seg	SparkFun Serial 7-Segment Display	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
thing	SparkFun ESP8266 Thing	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB
thingdev	SparkFun ESP8266 Thing Dev	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB
uvview	SparkFun MicroView	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB

SparkFun Electronics

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp32thing	SparkFun ESP32 Thing	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

SpellFoundry

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
sleepypi	SpellFoundry Sleepy Pi 2	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	30KB	2KB

SweetPea

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp210	SweetPea ESP-210	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

TTGO

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
ttgo-lora32-v1	TTGO LoRa32-OLED V1	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Taida Century

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
stct_nrf52_minidev	Taida Century nRF52 mini board	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

Teensy

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
teensy20	Teensy 2.0	<i>Teensy</i>	No	ATMEGA32U4	16MHz	31.50KB	2.50KB
teensy20pp	Teensy++ 2.0	<i>Teensy</i>	No	AT90USB1286	16MHz	127KB	8KB
teensy30	Teensy 3.0	<i>Teensy</i>	No	MK20DX128	48MHz	128KB	16KB
teensy31	Teensy 3.1 / 3.2	<i>Teensy</i>	<i>Yes</i>	MK20DX256	72MHz	256KB	64KB
teensy35	Teensy 3.5	<i>Teensy</i>	<i>Yes</i>	MK64FX512	120MHz	512KB	192KB
teensy36	Teensy 3.6	<i>Teensy</i>	<i>Yes</i>	MK66FX1M0	180MHz	1MB	256KB
teensylc	Teensy LC	<i>Teensy</i>	<i>Yes</i>	MKL26Z64	48MHz	62KB	8KB

ThaiEasyElec

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espino32	ESPino32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
espinotee	ThaiEasyElec ESPino	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

The Things Network

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
the_things_uno	The Things Uno	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB

TinyCircuits

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
tinyduino	TinyCircuits TinyDuino Processor Board	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	30KB	2KB
tinylily	TinyCircuits TinyLily Mini Processor	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	30KB	2KB

UBW32

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
ubw32_mx460	UBW32 MX460	<i>Microchip PIC32</i>	No	32MX460F512L	80MHz	508KB	32KB
ubw32_mx795	UBW32 MX795	<i>Microchip PIC32</i>	No	32MX795F512L	80MHz	508KB	128KB

WEMOS

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
d1	WEMOS D1 R1 (Retired)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
d1_mini	WeMos D1 R2 & mini	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
d1_mini_lite	WeMos D1 mini Lite	<i>Espressif 8266</i>	No	ESP8266	80MHz	1MB	80KB
d1_mini_pro	WeMos D1 mini Pro	<i>Espressif 8266</i>	No	ESP8266	80MHz	16MB	80KB
lolin32	WEMOS LOLIN32	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB
lolin_d32	WEMOS LOLIN D32	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB
lolin_d32_pro	WEMOS LOLIN D32 PRO	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB
wemosbat	WeMos WiFi & Bluetooth Battery	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

Waveshare

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
waveshare_ble400	Waveshare BLE400	Nordic nRF51	Yes	NRF51822	32MHz	256KB	32KB

Wicked Device

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
wildfirev2	Wicked Device WildFire V2	Atmel AVR	No	AT-MEGA1284P	16MHz	120.00KB	16KB
wildfirev3	Wicked Device WildFire V3	Atmel AVR	No	AT-MEGA1284P	16MHz	127KB	16KB

Widora

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
widora-air	Widora AIR	Espressif 32	No	ESP32	240MHz	16MB	320KB

XinaBox

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
xinabox_cw02	XinaBox CW02	Espressif 32	Yes	ESP32	240MHz	4MB	320KB

chipKIT

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lenny	chipKIT Lenny	Microchip PIC32	No	32MX270F256D	40MHz	120KB	32KB

element14

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
chipkit_pi	Element14 chipKIT Pi	Microchip PIC32	No	32MX250F128B	40MHz	120KB	32KB

makerlab.mx

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
altair	Altair	<i>Atmel AVR</i>	No	ATMEGA256RFR2	16MHz	248KB	32KB

ng-beacon

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
ng_beacon	ng-beacon	<i>Nordic nRF51</i>	Yes	NRF51822	32MHz	256KB	32KB

nicai-systems

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
bob3	nicai-systems BOB3 coding bot	<i>Atmel AVR</i>	No	AT-MEGA88	8MHz	8KB	1KB
nibo2	nicai-systems NIBO 2 robot	<i>Atmel AVR</i>	No	AT-MEGA128	16MHz	128KB	4KB
nibobee	nicai-systems NIBObee robot	<i>Atmel AVR</i>	No	AT-MEGA16	15MHz	16KB	1KB
nibobee_1284	nicai-systems NIBObee robot with Tuning Kit	<i>Atmel AVR</i>	No	AT-MEGA1284P	20MHz	128KB	16KB
niboburger	nicai-systems NIBO burger robot	<i>Atmel AVR</i>	No	AT-MEGA16	15MHz	16KB	1KB
niboburger_1284	nicai-systems NIBO burger robot with Tuning Kit	<i>Atmel AVR</i>	No	AT-MEGA1284P	20MHz	128KB	16KB

u-blox

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
nina_w10	u-blox NINA-W10 series	<i>Espressif 32</i>	No	ESP32	240MHz	2MB	320KB
ublox_evk_nina_b1	u-blox EVK-NINA-B1	<i>Nordic nRF52</i>	Yes	NRF52832	64MHz	512KB	64KB

ubIQio

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
ardhat	ubIQio Ardhat	<i>Atmel AVR</i>	No	ATMEGA328P	16MHz	31.50KB	2KB

1.11.2 ARTIK SDK

framework = artik-sdk

ARTIK SDK is a C/C++ SDK targeting Samsung ARTIK platforms. It exposes a set of APIs to ease up development of applications. These APIs cover hardware buses such as GPIO, SPI, I2C, UART, connectivity links like Wi-Fi, Bluetooth, Zigbee, and network protocols such as HTTP, Websockets, MQTT, and others.

For more detailed information please visit [vendor site](#).

Contents

- [Examples](#)
- [Platforms](#)
- [Boards](#)

Examples

- ARTIK SDK for Linux ARM

Platforms

Name	Description
Linux ARM	Linux ARM is a Unix-like and mostly POSIX-compliant computer operating system (OS) assembled under the model of free and open-source software development and distribution. Using host OS (Mac OS X, Linux ARM) you can build native application for Linux ARM platform.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

RushUp

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
kitra_520	RushUp Kitra 520	Linux ARM	No	EXYNOS3250	1000MHz	4GB	512MB

Samsung

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
artik_1020	Samsung ARTIK 1020	<i>Linux ARM</i>	No	EXYNOS5422	1500MHz	16GB	2GB
artik_520	Samsung ARTIK 520	<i>Linux ARM</i>	No	EXYNOS3250	1000MHz	4GB	512MB
artik_530	Samsung ARTIK 530	<i>Linux ARM</i>	No	S5P4418	1200MHz	4GB	512MB
artik_710	Samsung ARTIK 710	<i>Linux ARM</i>	No	S5P6818	1400MHz	4GB	1GB

1.11.3 CMSIS

framework = cmsis

The ARM Cortex Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for the Cortex-M processor series and specifies debugger interfaces. The CMSIS enables consistent and simple software interfaces to the processor for interface peripherals, real-time operating systems, and middleware. It simplifies software re-use, reducing the learning curve for new microcontroller developers and cutting the time-to-market for devices.

For more detailed information please visit [vendor site](#).

Contents

- *Debugging*
- *Examples*
- *Platforms*
- *Boards*

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Banks listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
cloud_jam	RushUp Cloud-JAM	ST STM32	ST-LINK (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32F401RE	8MHz	512KB	96KB
disco_f303	STM32F3DISCO	ST STM32	ST-LINK (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32F303VCT6	7MHz	256KB	48KB
disco_f407	STM32F4DISCO	ST STM32	ST-LINK (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32F407VCT6	8MHz	1MB	128KB
disco_l152	STM32LDISCO	ST STM32	ST-LINK (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32L152RE	8MHz	128KB	16KB
nucleo_f401re	Nucleo F401RE	ST STM32	ST-LINK (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32F401RE	8MHz	512KB	96KB

External Debug Tools

Banks listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
1bitsy_stm32f411B1	1Bitsy	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F415RGT6	8MHz	1MB	128KB
armstrap_eagle1024	1024strap Eagle 1024	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F417VCT6	8MHz	1MB	192KB
armstrap_eagle2048	2048strap Eagle 2048	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F427VIT6	8MHz	1.99MB	256KB
armstrap_eagle512	512strap Eagle 512	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F407VHT6	8MHz	512KB	192KB
bluepill_f103c8	BluePill F103C8	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F103C8T6	7MHz	64KB	20KB

Examples

- CMSIS for ST STM32

Platforms

Name	Description
ST STM32	The STM32 family of 32-bit Flash MCUs based on the ARM Cortex-M processor is designed to offer new degrees of freedom to MCU users. It offers a 32-bit product range that combines very high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

1BitSquared

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
1bitsy_stm32f415rgt	1Bitsy	ST STM32	Yes	STM32F415RG	168MHz	1MB	128KB

Armstrap

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
armstrap_eagle10	Armstrap Eagle 1024	ST STM32	Yes	STM32F417VGT	168MHz	1MB	192KB
armstrap_eagle20	Armstrap Eagle 2048	ST STM32	Yes	STM32F427VIT	168MHz	1.99MB	256KB
armstrap_eagle51	Armstrap Eagle 512	ST STM32	Yes	STM32F407VET	168MHz	512KB	192KB

Generic

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
bluepill_f103c8	BluePill F103C8	ST STM32	Yes	STM32F103C8T6	72MHz	64KB	20KB

RushUp

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
cloud_jam	RushUp Cloud-JAM	ST STM32	Yes	STM32F401RET6	84MHz	512KB	96KB

ST

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
disco_f303vc	ST STM32F3DISCOVERY	ST STM32	Yes	STM32F303VCT	72MHz	256KB	48KB
disco_f407vg	ST STM32F4DISCOVERY	ST STM32	Yes	STM32F407VGT	168MHz	1MB	128KB
disco_l152rb	ST STM32LDISCOVERY	ST STM32	Yes	STM32L152RBT	32MHz	128KB	16KB
nucleo_f401re	ST Nucleo F401RE	ST STM32	Yes	STM32F401RET6	84MHz	512KB	96KB

1.11.4 Energia

framework = energia

Energia Wiring-based framework enables pretty much anyone to start easily creating microcontroller-based projects and applications. Its easy-to-use libraries and functions provide developers of all experience levels to start blinking LEDs, buzzing buzzers and sensing sensors more quickly than ever before.

For more detailed information please visit [vendor site](#).

Contents

- *Debugging*
- *Examples*
- *Platforms*
- *Boards*

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Banks listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
1plm4f120h5	TI LaunchPad (Stellaris) w/ lm4f120 (80MHz)	TI TIVA	TI-ICDI (on-board)	LPLM4F120H5	80MHz	256KB	32KB
1pmmsp430f5	TI LaunchPad MSP-EXP430F5529LP	TI MSP430	MSP Debug (on-board)	MSP430F5529	16MHz	128KB	8KB
1pmmsp430fr4	TI LaunchPad MSP-EXP430FR4133LP	TI MSP430	MSP Debug (on-board)	MSP430FR4133	38MHz	15KB	2KB
1pmmsp430fr5	TI FraunchPad MSP-EXP430FR5739LP	TI MSP430	MSP Debug (on-board)	MSP430FR5739	16MHz	16KB	512B
1pmmsp430fr5	TI LaunchPad MSP-EXP430FR5969LP	TI MSP430	MSP Debug (on-board)	MSP430FR5969	98MHz	64KB	2KB
1pmmsp430fr6	TI LaunchPad MSP-EXP430FR6989LP	TI MSP430	MSP Debug (on-board)	MSP430FR6989	98MHz	127KB	2KB
1pmmsp430g25	TI LaunchPad MSP-EXP430G2553LP	TI MSP430	MSP Debug (on-board)	MSP430G2553	16MHz	16KB	512B
1ptm4c1230c3	TI LaunchPad (Tiva C) w/ tm4c123 (80MHz)	TI TIVA	TI-ICDI (on-board)	LPTM4C1230C3	80MHz	256KB	32KB
1ptm4c1294n	TI LaunchPad (Tiva C) w/ tm4c129 (120MHz)	TI TIVA	TI-ICDI (on-board)	LPTM4C1294N	120MHz	1MB	256KB

Examples

- Energia for TI MSP430
- Energia for TI TIVA

Platforms

Name	Description
TI MSP430	MSP430 microcontrollers (MCUs) from Texas Instruments (TI) are 16-bit, RISC-based, mixed-signal processors designed for ultra-low power. These MCUs offer the lowest power consumption and the perfect mix of integrated peripherals for thousands of applications.
TI TIVA	Texas Instruments TM4C12x MCUs offer the industry's most popular ARM Cortex-M4 core with scalable memory and package options, unparalleled connectivity peripherals, advanced application functions, industry-leading analog integration, and extensive software solutions.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

TI

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
1plm4f120h5q	TI LaunchPad (Stellaris) w/ lm4f120 (80MHz)	TI TIVA	Yes	LPLM4F120H5Q	80MHz	256KB	32KB
1pmmsp430f552	TI LaunchPad MSP-EXP430F5529LP	TI MSP430	Yes	MSP430F5529	16MHz	128KB	8KB
1pmmsp430fr413	TI LaunchPad MSP-EXP430FR4133LP	TI MSP430	Yes	MSP430FR4133	8MHz	15KB	2KB
1pmmsp430fr573	TI FraunchPad MSP-EXP430FR5739LP	TI MSP430	Yes	MSP430FR5739	16MHz	16KB	512B
1pmmsp430fr596	TI LaunchPad MSP-EXP430FR5969LP	TI MSP430	Yes	MSP430FR5969	8MHz	64KB	2KB
1pmmsp430fr698	TI LaunchPad MSP-EXP430FR6989LP	TI MSP430	Yes	MSP430FR6989	8MHz	127KB	2KB
1pmmsp430g255	TI LaunchPad MSP-EXP430G2553LP	TI MSP430	Yes	MSP430G2553	16MHz	16KB	512B
1ptm4c1230c3p	TI LaunchPad (Tiva C) w/ tm4c123 (80MHz)	TI TIVA	Yes	LPTM4C1230C3P	80MHz	256KB	32KB
1ptm4c1294ncp	TI LaunchPad (Tiva C) w/ tm4c129 (120MHz)	TI TIVA	Yes	LPTM4C1294NC	120MHz	1MB	256KB

1.11.5 ESP8266 Non-OS SDK

`framework = esp8266-nonos-sdk`

The non-OS SDK provides a set of application programming interfaces (APIs) for core ESP8266 functionalities such as data reception/transmission over Wi-Fi, TCP/IP stack functions, hardware interface functions and basic system management functions.

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Platforms*
- *Boards*

Examples

- ESP8266 Non-OS SDK for Espressif 8266

Platforms

Name	Description
<i>Espressif 8266</i>	Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

4D Systems

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
gen4iod	4D Systems gen4 IoD Range	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB

Adafruit

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
huzzah	Adafruit HUZZAH ESP8266	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Doit

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
espduino	ESPDuino (ESP-13 Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

DycodeX

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espectro	ESPectro Core	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

ESPert

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
espresso_lite_v1	ESPRESSO Lite 1.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
espresso_lite_v2	ESPRESSO Lite 2.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

ESPino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espino	ESPino	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Espressif

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
esp01	Espressif Generic ESP8266 ESP-01 512k	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB
esp01_1m	Espressif Generic ESP8266 ESP-01 1M	<i>Espressif 8266</i>	No	ESP8266	80MHz	1MB	80KB
esp07	Espressif Generic ESP8266 ESP-07	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
esp12e	Espressif ESP8266 ESP-12E	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
esp8285	Generic ESP8285 Module	<i>Espressif 8266</i>	No	ESP8266	80MHz	423.98KB	80KB
esp_wroom_02	ESP-WROOM-02	<i>Espressif 8266</i>	No	ESP8266	80MHz	2MB	80KB
phoenix_v1	Phoenix 1.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
phoenix_v2	Phoenix 2.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
wifinfo	WifInfo	<i>Espressif 8266</i>	No	ESP8266	80MHz	1MB	80KB

Heltec

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
heltec_wifi_kit_8	Heltec Wifi kit 8	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

NodeMCU

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
nodemcu	NodeMCU 0.9 (ESP-12 Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
nodemcuv2	NodeMCU 1.0 (ESP-12E Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Olimex

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
modwifi	Olimex MOD-WIFI-ESP8266(-DEV)	<i>Espressif 8266</i>	No	ESP8266	80MHz	2MB	80KB

SeeedStudio

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
wio_node	Wio Node	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

SparkFun

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
sparkfunBlynk	SparkFun Blynk Board	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
thing	SparkFun ESP8266 Thing	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB
thingdev	SparkFun ESP8266 Thing Dev	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB

SweetPea

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp210	SweetPea ESP-210	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

ThaiEasyElec

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espinotee	ThaiEasyElec ESPino	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

WEMOS

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
d1	WEMOS D1 R1 (Retired)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
d1_mini	WeMos D1 R2 & mini	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
d1_mini_pro	WeMos D1 mini Pro	<i>Espressif 8266</i>	No	ESP8266	80MHz	16MB	80KB

1.11.6 ESP8266 RTOS SDK

framework = esp8266-rtos-sdk

ESP8266 SDK based on FreeRTOS, a truly free professional grade RTOS for microcontrollers

For more detailed information please visit [vendor site](#).

Contents

- *Examples*
- *Platforms*
- *Boards*

Examples

- ESP8266 RTOS SDK for Espressif 8266

Platforms

Name	Description
<i>Espressif 8266</i>	Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

4D Systems

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
gen4iod	4D Systems gen4 IoD Range	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB

Adafruit

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
huzzah	Adafruit HUZZAH ESP8266	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Doit

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
espduino	ESPDuino (ESP-13 Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

DycodeX

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espectro	ESPectro Core	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

ESPert

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
espresso_lite_v1	ESPRESSO Lite 1.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
espresso_lite_v2	ESPRESSO Lite 2.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

ESPino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espino	ESPino	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Espressif

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
esp01	Espressif Generic ESP8266 ESP-01 512k	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB
esp01_1m	Espressif Generic ESP8266 ESP-01 1M	<i>Espressif 8266</i>	No	ESP8266	80MHz	1MB	80KB
esp07	Espressif Generic ESP8266 ESP-07	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
esp12e	Espressif ESP8266 ESP-12E	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
esp8285	Generic ESP8285 Module	<i>Espressif 8266</i>	No	ESP8266	80MHz	423.98KB	80KB
esp_wroom_02	ESP-WROOM-02	<i>Espressif 8266</i>	No	ESP8266	80MHz	2MB	80KB
phoenix_v1	Phoenix 1.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
phoenix_v2	Phoenix 2.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
wifinfo	WifInfo	<i>Espressif 8266</i>	No	ESP8266	80MHz	1MB	80KB

Heltec

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
heltec_wifi_kit_8	Heltec Wifi kit 8	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

NodeMCU

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
nodemcu	NodeMCU 0.9 (ESP-12 Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
nodemcuv2	NodeMCU 1.0 (ESP-12E Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Olimex

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
modwifi	Olimex MOD-WIFI-ESP8266(-DEV)	<i>Espressif 8266</i>	No	ESP8266	80MHz	2MB	80KB

SeeedStudio

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
wio_node	Wio Node	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

SparkFun

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
sparkfunBlynk	SparkFun Blynk Board	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
thing	SparkFun ESP8266 Thing	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB
thingdev	SparkFun ESP8266 Thing Dev	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB

SweetPea

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp210	SweetPea ESP-210	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

ThaiEasyElec

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espinotee	ThaiEasyElec ESPino	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

WEMOS

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
d1	WEMOS D1 R1 (Retired)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
d1_mini	WeMos D1 R2 & mini	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
d1_mini_pro	WeMos D1 mini Pro	<i>Espressif 8266</i>	No	ESP8266	80MHz	16MB	80KB

1.11.7 ESP-IDF

`framework = espidf`

Espressif IoT Development Framework. Official development framework for ESP32.

For more detailed information please visit [vendor site](#).

Contents

- [Debugging](#)
- [Examples](#)
- [Platforms](#)
- [Boards](#)

Debugging

[PIO Unified Debugger](#) - “1-click” solution for debugging with a zero configuration.

- [Debug Tools](#)
 - [On-Board Debug Tools](#)
 - [External Debug Tools](#)

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging [Tools & Debug Probes](#) using `debug_tool` options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Bands listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp-wrover-kit	Espressif ESP-WROVER-KIT	Espresso-FTDI Chip (default, on-board), Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY		ESP32	240MHz	4MB	320KB

External Debug Tools

Boards listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
esp32-evo	OLIMEX ESP32-EVB	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
esp32-gate	OLIMEX ESP32- GATEWAY	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
esp32dev	Espressif ESP32 Dev Module	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
esp32doit	DOIT ESP32 DEVKIT V1	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
esp32thing	SparkFun ESP32 Thing	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
esp32vn-i	ESP32n IoT Uno	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
espectro3	ESPectro32	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
espino32	ESPino32	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
featheresp32	A2fruit ESP32 Feather	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
firebeetle	FireBeetle- ESP32	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
hornbill32	Hornbill ESP32 Dev	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
hornbill32m	Hornbill ESP32 Minima	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
lolin32	WEMOS LOLIN32	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
lolin_d32	WEMOS LOLIN D32	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
lolin_d32p	WEMOS LOLIN D32 PRO	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
mhetesp32d	MHET LIVE ESP32DevKIT	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
mhetesp32m	MHET LIVE ESP32MiniKit	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
node32s	Node32s	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	Chapter 1. Contents
nodemcu-32S	NodeMCU- 32S	Espressif 32	-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	

Examples

- ESP-IDF for Espressif 32

Platforms

Name	Description
Espressif 32	Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

Adafruit

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
featheresp32	Adafruit ESP32 Feather	Espressif 32	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Aiyarafun

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
node32s	Node32s	Espressif 32	<i>Yes</i>	ESP32	240MHz	4MB	320KB

April Brother

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espea32	April Brother ESPea32	Espressif 32	No	ESP32	240MHz	4MB	320KB

DFRobot

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
firebeetle32	FireBeetle-ESP32	Espressif 32	<i>Yes</i>	ESP32	240MHz	4MB	320KB

DOIT

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
esp32doit-devkit-v1	DOIT ESP32 DEVKIT V1	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Dongsen Technology

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
pocket_32	Dongsen Tech Pocket 32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

DycodeX

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espectro32	ESPectro32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

ESP32vn

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp32vn-iot-uno	ESP32vn IoT Uno	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Electronic SweetPeas

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp320	Electronic SweetPeas ESP320	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

Espressif

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
esp-wrover-kit	Espressif ESP-WROVER-KIT	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
esp32dev	Espressif ESP32 Dev Module	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
pico32	ESP32 Pico Kit	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

Hardkernel

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
odroid_esp32	ODROID-GO	<i>Espressif 32</i>	No	ESP32	240MHz	16MB	320KB

Heltec Automation

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
heltec_wifi_kit_32	Heltec WIFI Kit 32	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB
heltec_wifi_lora_32	Heltec WIFI LoRa 32	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

Hornbill

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
hornbill132dev	Hornbill ESP32 Dev	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
hornbill132minima	Hornbill ESP32 Minima	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

IntoRobot

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
intorobot	IntoRobot Fig	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

M5Stack

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
m5stack-core-esp32	M5Stack Core	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB
m5stack-fire	M5Stack FIRE	<i>Espressif 32</i>	No	ESP32	240MHz	16MB	320KB

MH-ET Live

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
mhetesp32devkit	MH ET LIVE ESP32DevKIT	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
mhetesp32minikit	MH ET LIVE ESP32MiniKit	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

MakerAsia

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
nano32	MakerAsia Nano32	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

Microduino

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
microduino-core-esp32	Microduino Core	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

NodeMCU

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
nodemcu-32s	NodeMCU-32S	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Noduino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
quantum	Noduino Quantum	<i>Espressif 32</i>	No	ESP32	240MHz	16MB	320KB

OLIMEX

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
esp32-evb	OLIMEX ESP32-EVB	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
esp32-gateway	OLIMEX ESP32-GATEWAY	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Onehorse

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
onehorse32dev	Onehorse ESP32 Dev Mod-ule	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

SparkFun Electronics

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp32thing	SparkFun ESP32 Thing	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

TTGO

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
ttgo-lora32-v1	TTGO LoRa32-OLED V1	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

ThaiEasyElec

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espino32	ESPino32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

WEMOS

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
lolin32	WEMOS LOLIN32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
lolin_d32	WEMOS LOLIN D32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
lolin_d32_pro	WEMOS LOLIN D32 PRO	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
wemosbat	WeMos WiFi & Bluetooth Battery	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

Widora

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
widora-air	Widora AIR	<i>Espressif 32</i>	No	ESP32	240MHz	16MB	320KB

XinaBox

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
xinabox_cw02	XinaBox CW02	Espressif 32	Yes	ESP32	240MHz	4MB	320KB

u-blox

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
nina_w10	u-blox NINA-W10 series	Espressif 32	No	ESP32	240MHz	2MB	320KB

1.11.8 Freedom E SDK

framework = freedom-e-sdk

Open Source Software for Developing on the SiFive Freedom E Platform

For more detailed information please visit [vendor site](#).

Contents

- *Debugging*
- *Examples*
- *Platforms*
- *Boards*

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Boards listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
freedom-e300-hifive1	HiFive1	RISC-V	FTDI Chip (on-board)	FE310	320MHz	16MB	16KB

External Debug Tools

Boards listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
coreplexip-e31	Freedom E310 Arty (Artix-7) FPGA Dev Kit	RISC-V	Olimex ARM-USB-TINY-H	E31	320MHz	16MB	256MB
coreplexip-e51	Arty Arty (Artix-7) FPGA Dev Kit	RISC-V	Olimex ARM-USB-TINY-H	E51	1500MHz	16MB	256MB

Examples

- Freedom E SDK for RISC-V

Platforms

Name	Description
RISC-V	RISC-V is an open, free ISA enabling a new era of processor innovation through open standard collaboration. Born in academia and research, RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.

Boards

Note:

- You can list pre-configured boards by *platformio boards* command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

SiFive

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
freedom-e300-hifive1	HiFive1	RISC-V	Yes	FE310	320MHz	16MB	16KB

Xilinx

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
coreplexip-e31-artix7	Freedom E310 Arty (Artix-7) FPGA Dev Kit	RISC-V	Yes	E31	320MHz	16MB	256MB
coreplexip-e51-artix7	RT51 Arty (Artix-7) FPGA Dev Kit	RISC-V	Yes	E51	1500MHz	16MB	256MB

1.11.9 libOpenCM3

`framework = libopencm3`

The libOpenCM3 framework aims to create a free/libre/open-source firmware library for various ARM Cortex-M0(+)/M3/M4 microcontrollers, including ST STM32, TI Tiva and Stellaris, NXP LPC 11xx, 13xx, 15xx, 17xx parts, Atmel SAM3, Energy Micro EFM32 and others.

For more detailed information please visit [vendor site](#).

Contents

- [Debugging](#)
- [Examples](#)
- [Platforms](#)
- [Boards](#)

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- [Debug Tools](#)
 - [On-Board Debug Tools](#)
 - [External Debug Tools](#)

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging [Tools & Debug Probes](#) using `debug_tool` options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Boards listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
disco_f30	STM32F3DISCOVERY	ST STM32	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i> , <i>J-LINK</i>	STM32F303VCT6	72MHz	256KB	48KB
disco_f40	STM32F4DISCOVERY	ST STM32	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i> , <i>J-LINK</i>	STM32F407VCT6	88MHz	1MB	128KB
disco_l15	STM32LDISCOVERY	ST STM32	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i> , <i>J-LINK</i>	STM32L152RB	32MHz	128KB	16KB
lplm4f120	TI LaunchPad (Stellaris) w/ lm4f120 (80MHz)	TI TIVA	<i>TI-ICDI</i> (on-board)	LPLM4F120HSQ	80MHz	256KB	32KB
lptm4c123	TI LaunchPad (Tiva C) w/ tm4c123 (80MHz)	TI TIVA	<i>TI-ICDI</i> (on-board)	LPTM4C123080MM	80MHz	256KB	32KB
lptm4c129	TI LaunchPad (Tiva C) w/ tm4c129 (120MHz)	TI TIVA	<i>TI-ICDI</i> (on-board)	LPTM4C1294NOMDIE	120MHz	1MB	256KB
nucleo_f103rb	Nucleo F103RB	ST STM32	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i> , <i>J-LINK</i>	STM32F103RB	72MHz	128KB	20KB

External Debug Tools

Boards listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
1bitsy_stm32f411e	1BitSugt	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F411RE	8MHz	1MB	128KB
bluepill_f103	BluePill F103C8	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CB	8MHz	64KB	20KB
genericSTM32F103C8	(20k RAM. 64k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103C8	8MHz	64KB	20KB
genericSTM32F103CB	(20k RAM. 128k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CB	8MHz	128KB	20KB
genericSTM32F103R8	(20k RAM. 64k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103R8	8MHz	64KB	20KB
genericSTM32F103RB	(20k RAM. 128k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103RB	8MHz	128KB	20KB
genericSTM32F103RC	(48k RAM. 256k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103RC	8MHz	256KB	48KB
genericSTM32F103RE	(64k RAM. 512k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103RE	8MHz	512KB	64KB
genericSTM32F103VC	(48k RAM. 256k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103VC	8MHz	256KB	48KB
genericSTM32F103VE	(64k RAM. 512k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103VE	8MHz	512KB	64KB
maple	Maple	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103RA	8MHz	108KB	17KB
maple_mini_b2	Maple Mini Boot-loader 2.0	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CB	8MHz	120KB	20KB
maple_mini_cr	Maple Mini Original	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CE	8MHz	108KB	17KB

Examples

- libOpenCM3 for ST STM32
- libOpenCM3 for TI TIVA

Platforms

Name	Description
ST STM32	The STM32 family of 32-bit Flash MCUs based on the ARM Cortex-M processor is designed to offer new degrees of freedom to MCU users. It offers a 32-bit product range that combines very high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.
TI TIVA	Texas Instruments TM4C12x MCUs offer the industry's most popular ARM Cortex-M4 core with scalable memory and package options, unparalleled connectivity peripherals, advanced application functions, industry-leading analog integration, and extensive software solutions.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

1BitSquared

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
1bitsy_stm32f415rgt	1Bitsy	ST STM32	Yes	STM32F415RGT	168MHz	1MB	128KB

Generic

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
bluepill_f103c8	BluePill F103C8	ST STM32	Yes	STM32F103C8T62MHz	64KB	20KB	
genericSTM32F103C8	STM32F103C8 (20k RAM. 64k Flash)	ST STM32	Yes	STM32F103C8T62MHz	64KB	20KB	
genericSTM32F103CB	STM32F103CB (20k RAM. 128k Flash)	ST STM32	Yes	STM32F103CBT62MHz	128KB	20KB	
genericSTM32F103R8	STM32F103R8 (20k RAM. 64 Flash)	ST STM32	Yes	STM32F103R8T62MHz	64KB	20KB	
genericSTM32F103RB	STM32F103RB (20k RAM. 128k Flash)	ST STM32	Yes	STM32F103RBT62MHz	128KB	20KB	
genericSTM32F103RC	STM32F103RC (48k RAM. 256k Flash)	ST STM32	Yes	STM32F103RCT62MHz	256KB	48KB	
genericSTM32F103RE	STM32F103RE (64k RAM. 512k Flash)	ST STM32	Yes	STM32F103RET62MHz	512KB	64KB	
genericSTM32F103VC	STM32F103VC (48k RAM. 256k Flash)	ST STM32	Yes	STM32F103VCT62MHz	256KB	48KB	
genericSTM32F103VE	STM32F103VE (64k RAM. 512k Flash)	ST STM32	Yes	STM32F103VET62MHz	512KB	64KB	

LeafLabs

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
maple	Maple	ST STM32	Yes	STM32F103RBT62MHz	108KB	17KB	
maple_mini_b20	Maple Mini Boot-loader 2.0	ST STM32	Yes	STM32F103CBT62MHz	120KB	20KB	
maple_mini_orig	Maple Mini Original	ST STM32	Yes	STM32F103CBT62MHz	108KB	17KB	

ST

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
disco_f303vc	ST STM32F3DISCOVERY	ST STM32	Yes	STM32F303VCT6	72MHz	256KB	48KB
disco_f407vg	ST STM32F4DISCOVERY	ST STM32	Yes	STM32F407VGT6	168MHz	1MB	128KB
disco_l152rb	ST STM32LDISCOVERY	ST STM32	Yes	STM32L152RBT6	32MHz	128KB	16KB
nucleo_f103rb	ST Nucleo F103RB	ST STM32	Yes	STM32F103RBT6	72MHz	128KB	20KB

TI

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
lplm4f120h5q	TI LaunchPad (Stellaris) w/ lm4f120 (80MHz)	TI TIVA	Yes	LPLM4F120H5Q	80MHz	256KB	32KB
lptm4c1230c3p	TI LaunchPad (Tiva C) w/ tm4c123 (80MHz)	TI TIVA	Yes	LPTM4C1230C3P	80MHz	256KB	32KB
lptm4c1294ncp	TI LaunchPad (Tiva C) w/ tm4c129 (120MHz)	TI TIVA	Yes	LPTM4C1294NC	120MHz	1MB	256KB

1.11.10 mbed

framework = mbed

The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

For more detailed information please visit [vendor site](#).

Contents

- [Configuration](#)
- [Debugging](#)
- [Examples](#)
- [Platforms](#)
- [Boards](#)

Configuration

- *RTOS, Events, FileSystem*
- *Ignore built-in libraries*

RTOS, Events, FileSystem

mbed framework consists of several components, some of which should be explicitly added to the build process. For this purpose PlatformIO has special macro definitions that should be added to *build_flags* of *Project Configuration File platformio.ini* when one of the components is used in a project:

Name	Description
PIO_FRAMEWORKMBED_RTOS_PRESENT	Build the project with enabled <code>rtos</code> library
PIO_FRAMEWORKMBED_EVENTS_PRESENT	Build the project with enabled <code>events</code> library
PIO_FRAMEWORKMBED_FILESYSTEM_PRESENT	Build the project with enabled <code>filesystem</code> library

An example of *Project Configuration File platformio.ini* with enabled `events` library

```
[env:wizwiki_w7500p]
platform = wiznet7500
framework = mbed
board = wizwiki_w7500p
build_flags = -D PIO_FRAMEWORKMBED_EVENTS_PRESENT
```

An example of *Project Configuration File platformio.ini* with `events` and `rtos` libraries

```
[env:nrf52_dk]
platform = nordicnrf52
framework = mbed
board = nrf52_dk
build_flags = -D PIO_FRAMEWORKMBED_EVENTS_PRESENT -D PIO_FRAMEWORKMBED_RTOS_PRESENT
```

An example of *Project Configuration File platformio.ini* with `filesystem` library

```
[env:nucleo_f767zi]
platform = ststm32
framework = mbed
board = nucleo_f767zi
build_flags = -D PIO_FRAMEWORKMBED_FILESYSTEM_PRESENT
```

Ignore built-in libraries

See `lib_ignore`.

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*

– *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Banks listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Platform	Debug
IBMEthernetKit	Ethernet IoT Starter Kit	Freescale Kinetis	CMSIS
b96b_f446ve	96Boards B96B-F446VE	ST STM32	ST-LIN
bbcmicrobit	BBC micro:bit	Nordic nRF51	CMSIS
bbcmicrobit_b	BBC micro:bit B(S130)	Nordic nRF51	CMSIS
cloud_jam	RushUp Cloud-JAM	ST STM32	ST-LIN
cloud_jam_14	RushUp Cloud-JAM L4	ST STM32	ST-LIN
delta_dfbm_nq620	Delta DFBM-NQ620	Nordic nRF52	CMSIS
delta_dfcm_nnn50	Delta DFMC-NNN50	Nordic nRF51	CMSIS
dfcm_nnn40	Delta DFMC-NNN40	Nordic nRF51	CMSIS
disco_f030r8	ST STM32F0308DISCOVERY	ST STM32	ST-LIN
disco_f051r8	ST STM32F0DISCOVERY	ST STM32	ST-LIN
disco_f100rb	ST STM32VLDISCOVERY	ST STM32	ST-LIN
disco_f303vc	ST STM32F3DISCOVERY	ST STM32	ST-LIN
disco_f334c8	ST 32F3348DISCOVERY	ST STM32	ST-LIN
disco_f401vc	ST 32F401CDISCOVERY	ST STM32	ST-LIN
disco_f407vg	ST STM32F4DISCOVERY	ST STM32	ST-LIN
disco_f413zh	ST 32F413HDISCOVERY	ST STM32	ST-LIN
disco_f429zi	ST 32F429IDISCOVERY	ST STM32	ST-LIN
disco_f469ni	ST 32F469IDISCOVERY	ST STM32	ST-LIN
disco_f746ng	ST 32F746GDISCOVERY	ST STM32	ST-LIN
disco_f769ni	ST 32F769IDISCOVERY	ST STM32	ST-LIN
disco_l053c8	ST 32L0538DISCOVERY	ST STM32	ST-LIN
disco_l072cz_lrwan1	ST DISCO-L072CZ-LRWAN1	ST STM32	ST-LIN
disco_l475vg_iot01a	ST DISCO-L475VG-IOT01A	ST STM32	ST-LIN
disco_l476vg	ST 32L476GDISCOVERY	ST STM32	ST-LIN
efm32gg_stk3700	Silicon Labs EFM32GG-STK3700 (Giant Gecko)	Silicon Labs EFM32	J-LINK
efm32hg_stk3400	Silicon Labs SLSTK3400A USB-enabled (Happy Gecko)	Silicon Labs EFM32	J-LINK
efm32lg_stk3600	Silicon Labs EFM32LG-STK3600 (Leopard Gecko)	Silicon Labs EFM32	J-LINK
efm32pg_stk3401	Silicon Labs SLSTK3401A (Pearl Gecko)	Silicon Labs EFM32	J-LINK
efm32wg_stk3800	Silicon Labs EFM32WG-STK3800 (Wonder Gecko)	Silicon Labs EFM32	J-LINK
efm32zg_stk3200	Silicon Labs EFM32ZG-STK3200 (Zero Gecko)	Silicon Labs EFM32	J-LINK

Table 7 – continued from page 1

ID	Name	Platform	Debug
elektor_cocorico	CoCo-ri-Co!	NXP LPC	CMSIS-DK
frdm_k20d50m	Freescale Kinetis FRDM-K20D50M	Freescale Kinetis	CMSIS-DK
frdm_k22f	Freescale Kinetis FRDM-K22F	Freescale Kinetis	CMSIS-DK
frdm_k64f	Freescale Kinetis FRDM-K64F	Freescale Kinetis	CMSIS-DK
frdm_k66f	Freescale Kinetis FRDM-K66F	Freescale Kinetis	CMSIS-DK
frdm_kl05z	Freescale Kinetis FRDM-KL05Z	Freescale Kinetis	CMSIS-DK
frdm_kl25z	Freescale Kinetis FRDM-KL25Z	Freescale Kinetis	CMSIS-DK
frdm_kl27z	Freescale Kinetis FRDM-KL27Z	Freescale Kinetis	CMSIS-DK
frdm_kl43z	Freescale Kinetis FRDM-KL43Z	Freescale Kinetis	CMSIS-DK
frdm_kl46z	Freescale Kinetis FRDM-KL46Z	Freescale Kinetis	CMSIS-DK
frdm_kw41z	Freescale Kinetis FRDM-KW41Z	Freescale Kinetis	CMSIS-DK
hrm1017	Switch Science mbed HRM1017	Nordic nRF51	CMSIS-DK
lpc1114fn28	Switch Science mbed LPC1114FN28	NXP LPC	CMSIS-DK
lpc11u24	NXP mbed LPC11U24	NXP LPC	CMSIS-DK
lpc11u24_301	ARM mbed LPC11U24 (+CAN)	NXP LPC	CMSIS-DK
lpc11u68	LPCXpresso11U68	NXP LPC	CMSIS-DK
lpc1768	NXP mbed LPC1768	NXP LPC	CMSIS-DK
lpc4088	Embedded Artists LPC4088 QuickStart Board	NXP LPC	CMSIS-DK
lpc4088_dm	Embedded Artists LPC4088 Display Module	NXP LPC	CMSIS-DK
lpc4330_m4	Bambino-210E	NXP LPC	CMSIS-DK
lpc54114	NXP LPCXpresso54114	NXP LPC	CMSIS-DK
lpc546xx	NXP LPCXpresso54608	NXP LPC	J-LINK
lpc812	NXP LPC800-MAX	NXP LPC	CMSIS-DK
lpc824	LPCXpresso824-MAX	NXP LPC	CMSIS-DK
max32600mbed	Maxim ARM mbed Enabled Development Platform for MAX32600	Maxim 32	CMSIS-DK
mbed_connect_odin	Mbed Connect Cloud	ST STM32	CMSIS-DK
nrf51822_y5_mbug	y5 nRF51822 mbug	Nordic nRF51	CMSIS-DK
nrf51_dk	Nordic nRF51-DK	Nordic nRF51	CMSIS-DK
nrf51_dongle	Nordic nRF51-Dongle	Nordic nRF51	CMSIS-DK
nrf51_mkit	Nordic nRF51822-mKIT	Nordic nRF51	CMSIS-DK
nrf52840_dk	Nordic nRF52840-DK	Nordic nRF52	CMSIS-DK
nrf52_dk	Nordic nRF52-DK	Nordic nRF52	CMSIS-DK
nucleo_f030r8	ST Nucleo F030R8	ST STM32	ST-LINK
nucleo_f031k6	ST Nucleo F031K6	ST STM32	ST-LINK
nucleo_f042k6	ST Nucleo F042K6	ST STM32	ST-LINK
nucleo_f070rb	ST Nucleo F070RB	ST STM32	ST-LINK
nucleo_f072rb	ST Nucleo F072RB	ST STM32	ST-LINK
nucleo_f091rc	ST Nucleo F091RC	ST STM32	ST-LINK
nucleo_f103rb	ST Nucleo F103RB	ST STM32	ST-LINK
nucleo_f207zg	ST Nucleo F207ZG	ST STM32	ST-LINK
nucleo_f302r8	ST Nucleo F302R8	ST STM32	ST-LINK
nucleo_f303k8	ST Nucleo F303K8	ST STM32	ST-LINK
nucleo_f303re	ST Nucleo F303RE	ST STM32	ST-LINK
nucleo_f303ze	ST Nucleo F303ZE	ST STM32	ST-LINK
nucleo_f334r8	ST Nucleo F334R8	ST STM32	ST-LINK
nucleo_f401re	ST Nucleo F401RE	ST STM32	ST-LINK
nucleo_f410rb	ST Nucleo F410RB	ST STM32	ST-LINK
nucleo_f411re	ST Nucleo F411RE	ST STM32	ST-LINK
nucleo_f412zg	ST Nucleo F412ZG	ST STM32	ST-LINK

Table 7 – continued from page 39

ID	Name	Platform	Debug
nucleo_f413zh	ST Nucleo F413ZH	ST STM32	ST-LINK
nucleo_f429zi	ST Nucleo F429ZI	ST STM32	ST-LINK
nucleo_f446re	ST Nucleo F446RE	ST STM32	ST-LINK
nucleo_f446ze	ST Nucleo F446ZE	ST STM32	ST-LINK
nucleo_f746zg	ST Nucleo F746ZG	ST STM32	ST-LINK
nucleo_f767zi	ST Nucleo F767ZI	ST STM32	ST-LINK
nucleo_l011k4	ST Nucleo L011K4	ST STM32	ST-LINK
nucleo_l031k6	ST Nucleo L031K6	ST STM32	ST-LINK
nucleo_l053r8	ST Nucleo L053R8	ST STM32	ST-LINK
nucleo_l073rz	ST Nucleo L073RZ	ST STM32	ST-LINK
nucleo_l1152re	ST Nucleo L1152RE	ST STM32	ST-LINK
nucleo_l432kc	ST Nucleo L432KC	ST STM32	ST-LINK
nucleo_l476rg	ST Nucleo L476RG	ST STM32	ST-LINK
nucleo_l496zg	ST Nucleo L496ZG	ST STM32	ST-LINK
redBearLab	RedBearLab nRF51822	Nordic nRF51	CMSIS-DSP
redBearLabBLENano	RedBearLab BLE Nano 1.5	Nordic nRF51	CMSIS-DSP
samd21_xpro	Atmel SAMD21-XPRO	Atmel SAM	CMSIS-DSP
samd21g18a	Atmel ATSAMW25-XPRO	Atmel SAM	CMSIS-DSP
saml21_xpro_b	Atmel SAML21-XPRO-B	Atmel SAM	CMSIS-DSP
samr21_xpro	Atmel ATSAMR21-XPRO	Atmel SAM	CMSIS-DSP
seeedArchBLE	Seeed Arch BLE	Nordic nRF51	CMSIS-DSP
seeedArchLink	Seeed Arch Link	Nordic nRF51	CMSIS-DSP
seeedArchMax	Seeed Arch Max	ST STM32	ST-LINK
seeedArchPro	Seeed Arch Pro	NXP LPC	CMSIS-DSP
seeedTinyBLE	Seeed Tiny BLE	Nordic nRF51	CMSIS-DSP
ssci824	Switch Science mbed LPC824	NXP LPC	CMSIS-DSP
ty51822r3	Switch Science mbed TY51822r3	Nordic nRF51	CMSIS-DSP
ublox_evk_nina_b1	u-blox EVK-NINA-B1	Nordic nRF52	J-LINK
ubloxc027	u-blox C027	NXP LPC	CMSIS-DSP
vbluno51	VNG VBLUNO51	Nordic nRF51	CMSIS-DSP
wallbot_ble	JKSoft Wallbot BLE	Nordic nRF51	CMSIS-DSP
wizwiki_w7500	WIZwiki-W7500	WIZNet W7500	CMSIS-DSP
wizwiki_w7500eco	WIZwiki-W7500ECO	WIZNet W7500	CMSIS-DSP
wizwiki_w7500p	WIZwiki-W7500P	WIZNet W7500	CMSIS-DSP

External Debug Tools

Banks listed below are compatible with [PIO Unified Debugger](#) but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
blueboard_lp	NGXu24 Technologies BlueBoard-LPC11U24	NXP LPC	Black Magic Probe, J-LINK	LPC11U24	48MHz	32KB	8KB
bluepill_f10	Bla8Pill F103C8	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103C8M	48MHz	64KB	20KB
dipcortexm0	Solder Splash Labs DipCortex M0	NXP LPC	Black Magic Probe, J-LINK	LPC11U24	50MHz	32KB	8KB
elmo_f411re	Espotel LoRa Module	ST STM32	Black Magic Probe, J-LINK, ST-LINK (default)	STM32F411RE	72MHz	512KB	128KB
genericSTM32F103C8	STM32F103C8 (20k RAM, 64k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103C8M	48MHz	64KB	20KB
genericSTM32F103RB	STM32F103RB (20k RAM, 128k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103RB	72MHz	128KB	20KB
hexiwear	Hexiwear	Freescale Kinetis	CMSIS-DAP, J-LINK	MK64FN1M0VDMH	12MHz	1MB	256KB
lpc11c24	NXP LPC11C24	NXP LPC	Black Magic Probe, J-LINK	LPC11C24	48MHz	32KB	8KB
lpc11u34_421	NXP LPC11U34	NXP LPC	Black Magic Probe, J-LINK	LPC11U34	48MHz	40KB	8KB
lpc11u35	EA LPC11U35 QuickStart Board	NXP LPC	Black Magic Probe, J-LINK	LPC11U35	48MHz	64KB	10KB
lpc11u35_501	ICQ Publishing TG-LPC11U35-501	NXP LPC	Black Magic Probe, J-LINK	LPC11U35	48MHz	64KB	10KB
lpc11u35_y51	Y51 LPC11U35 mbug	NXP LPC	Black Magic Probe, J-LINK	LPC11U35	48MHz	64KB	10KB
lpc11u37_501	NXP LPC11U37	NXP LPC	Black Magic Probe, J-LINK	LPC11U37	48MHz	128KB	10KB
lpc1347	DipCortex M3	NXP LPC	J-LINK	LPC1347	72MHz	64KB	12KB
lpc1549	NXP LPCXpresso1549	NXP LPC	Black Magic Probe, J-LINK	LPC1549	72MHz	256KB	36KB
maxwsnenv	Maxim Wireless Sensor Node Demonstrator	Maxim 32	CMSIS-DAP	MAX32610	24MHz	256KB	32KB
mote_l1152rc	NAMote72	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32L152RC	48MHz	256KB	32KB
mtb_ublox_adihlow2	ODIN-W2	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F439ZI	88MHz	2MB	256KB
mts_dragonfly	MTS4 Dragonfly	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F411RE	72MHz	512KB	128KB
mts_mdot_f40	MultiTech mDot	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F411RE	72MHz	512KB	128KB
mts_mdot_f41	MultiTech mDot F411	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F411RE	72MHz	512KB	128KB
teensy31	Teensy 3.1 / 3.2	Teensy	J-LINK	MK20DX256	672MHz	256KB	64KB
ublox_c030_n2	uBlox C030-N211 IoT Starter Kit	ST STM32	Black Magic Probe, CMSIS-DAP, J-LINK, ST-LINK	STM32F437VI	80MHz	1MB	256KB
ublox_c030_u2	uBlox C030-U201 IoT Starter Kit	ST STM32	Black Magic Probe, CMSIS-DAP, J-LINK, ST-LINK	STM32F437VI	80MHz	1MB	256KB
ublox_evk_adihlow2	EVK-ODIN-W2	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F439ZI	88MHz	2MB	256KB
1.11. Frameworks							407
xdot_1151cc	MultiTech xDot	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32L151C2M	32MHz	256KB	32KB

Examples

- mbed for Atmel SAM
- mbed for Freescale Kinetis
- mbed for Maxim 32
- mbed for Nordic nRF51
- mbed for Nordic nRF52
- mbed for NXP LPC
- mbed for Silicon Labs EFM32
- mbed for ST STM32
- mbed for Teensy
- mbed for WIZNet W7500

Platforms

Name	Description
Atmel SAM	Atmel SMART offers Flash- based ARM products based on the ARM Cortex-M0+, Cortex-M3 and Cortex-M4 architectures, ranging from 8KB to 2MB of Flash including a rich peripheral and feature mix.
Freescale Kinetis	Freescale Kinetis Microcontrollers is family of multiple hardware- and software-compatible ARM Cortex-M0+, Cortex-M4 and Cortex-M7-based MCU series. Kinetis MCUs offer exceptional low-power performance, scalability and feature integration.
Maxim 32	Maxim's microcontrollers provide low-power, efficient, and secure solutions for challenging embedded applications. Maxim's processors embed cutting-edge technologies to secure data and intellectual property, proven analog circuitry for real-world applications, and battery-conserving low power operation.
Nordic nRF51	The Nordic nRF51 Series is a family of highly flexible, multi-protocol, system-on-chip (SoC) devices for ultra-low power wireless applications. nRF51 Series devices support a range of protocol stacks including Bluetooth Smart (previously called Bluetooth low energy), ANT and proprietary 2.4GHz protocols such as Gazell.
Nordic nRF52	The nRF52 Series are built for speed to carry out increasingly complex tasks in the shortest possible time and return to sleep, conserving precious battery power. They have a Cortex-M4F processor and are the most capable Bluetooth Smart SoCs on the market.
NXP LPC	The NXP LPC is a family of 32-bit microcontroller integrated circuits by NXP Semiconductors. The LPC chips are grouped into related series that are based around the same 32-bit ARM processor core, such as the Cortex-M4F, Cortex-M3, Cortex-M0+, or Cortex-M0. Internally, each microcontroller consists of the processor core, static RAM memory, flash memory, debugging interface, and various peripherals.
Silicon Labs EFM32	Silicon Labs EFM32 Gecko 32-bit microcontroller (MCU) family includes devices that offer flash memory configurations up to 256 kB, 32 kB of RAM and CPU speeds up to 48 MHz. Based on the powerful ARM Cortex-M core, the Gecko family features innovative low energy techniques, short wake-up time from energy saving modes and a wide selection of peripherals, making it ideal for battery operated applications and other systems requiring high performance and low-energy consumption.
ST STM32	The STM32 family of 32-bit Flash MCUs based on the ARM Cortex-M processor is designed to offer new degrees of freedom to MCU users. It offers a 32-bit product range that combines very high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.
Teensy	Teensy is a complete USB-based microcontroller development system, in a very small footprint, capable of implementing many types of projects. All programming is done via the USB port. No special programmer is needed, only a standard USB cable and a PC or Macintosh with a USB port.
WIZ-Net W7500	The IOP (Internet Offload Processor) W7500 is the one-chip solution which integrates an ARM Cortex-M0, 128KB Flash and hardwired TCP/IP core for various embedded application platform especially requiring Internet of things

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

96Boards

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
b96b_f446ve	96Boards B96B-F446VE	ST STM32	Yes	STM32F446VET6	168MHz	512KB	128KB

AppNearMe

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
micronfcboard	MicroNFCBoard	NXP LPC	No	LPC11U34	48MHz	48KB	10KB

Atmel

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
samd21_xpro	Atmel SAMD21-XPRO	Atmel SAM	Yes	SAMD21J18A	48MHz	256KB	32KB
samd21g18a	Atmel ATSAMW25-XPRO	Atmel SAM	Yes	SAMD21G18A	48MHz	256KB	32KB
saml21_xpro_b	Atmel SAML21-XPRO-B	Atmel SAM	Yes	SAML21J18B	48MHz	256KB	32KB
samr21_xpro	Atmel ATSAMR21-XPRO	Atmel SAM	Yes	SAMR21G18A	48MHz	256KB	32KB

BBC

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
bbcmicrobit	BBC micro:bit	Nordic nRF51	Yes	NRF51822	16MHz	256KB	16KB
bbcmicrobit_b	BBC micro:bit B(S130)	Nordic nRF51	Yes	NRF51822	16MHz	256KB	16KB

CQ Publishing

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lpc11u35_501	CQ Publishing TG-LPC11U35-501	NXP LPC	Yes	LPC11U35	48MHz	64KB	10KB

Delta

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
delta_dfbm_nq620	Delta DFBM-NQ620	Nordic nRF52	Yes	NRF52832	64MHz	512KB	64KB
delta_dfcn_nnn50	Delta DFCM-NNN50	Nordic nRF51	Yes	NRF51822	32MHz	256KB	16KB
dfcm_nnn40	Delta DFCM-NNN40	Nordic nRF51	Yes	NRF51822	32MHz	256KB	32KB

Elektor Labs

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
elektor_cocorico	CoCo-ri-Co!	NXP LPC	Yes	LPC812	30MHz	16KB	4KB

Embedded Artists

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
lpc11u35	EA LPC11U35 QuickStart Board	NXP LPC	Yes	LPC11U35	48MHz	64KB	10KB
lpc4088	Embedded Artists LPC4088 Quick-Start Board	NXP LPC	Yes	LPC4088	120MHz	512KB	96KB
lpc4088_dm	Embedded Artists LPC4088 Display Module	NXP LPC	Yes	LPC4088	120MHz	512KB	96KB

Espotel

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
elmo_f411re	Espotel LoRa Module	ST STM32	Yes	STM32F411RET6	100MHz	512KB	128KB

Freescale

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
IBMEthernet	KEthernet IoT Starter Kit	<i>Freescale Kinetis</i>	Yes	MK64FN1M0VLL120MHz	1MB	256KB	
frdm_k20d50m	Freescale Kinetis FRDM-K20D50M	<i>Freescale Kinetis</i>	Yes	MK20DX128VLH48MHz	128KB	16KB	
frdm_k22f	Freescale Kinetis FRDM-K22F	<i>Freescale Kinetis</i>	Yes	MK22FN512VLH1120MHz	512KB	128KB	
frdm_k64f	Freescale Kinetis FRDM-K64F	<i>Freescale Kinetis</i>	Yes	MK64FN1M0VLL120MHz	1MB	256KB	
frdm_k66f	Freescale Kinetis FRDM-K66F	<i>Freescale Kinetis</i>	Yes	MK66FN2M0VMD80MHz	2MB	256KB	
frdm_kl05z	Freescale Kinetis FRDM-KL05Z	<i>Freescale Kinetis</i>	Yes	MKL05Z32VFM448MHz	32KB	4KB	
frdm_kl25z	Freescale Kinetis FRDM-KL25Z	<i>Freescale Kinetis</i>	Yes	MKL25Z128VLK448MHz	128KB	16KB	
frdm_kl27z	Freescale Kinetis FRDM-KL27Z	<i>Freescale Kinetis</i>	Yes	MKL27Z64VLH448MHz	64KB	16KB	
frdm_kl43z	Freescale Kinetis FRDM-KL43Z	<i>Freescale Kinetis</i>	Yes	MKL43Z256VLH448MHz	256KB	32KB	
frdm_kl46z	Freescale Kinetis FRDM-KL46Z	<i>Freescale Kinetis</i>	Yes	MKL46Z256VLL448MHz	256KB	32KB	
frdm_kw41z	Freescale Kinetis FRDM-KW41Z	<i>Freescale Kinetis</i>	Yes	MKW41Z512VHT48MHz	512KB	128KB	

GHI Electronics

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
oc_mbuino	mBuino	<i>NXP LPC</i>	No	LPC11U24	50MHz	32KB	10KB

Generic

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
bluepill_f103c8	BluePill F103C8	<i>ST STM32</i>	Yes	STM32F103C8T6MHz	64KB	20KB	
genericSTM32F103C8	STM32F103C8 (20k RAM. 64k Flash)	<i>ST STM32</i>	Yes	STM32F103C8T6MHz	64KB	20KB	
genericSTM32F103RB	STM32F103RB (20k RAM. 128k Flash)	<i>ST STM32</i>	Yes	STM32F103RB70MHz	128KB	20KB	

JKSoft

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
wallbot_ble	JKSoft Wallbot BLE	Nordic nRF51	Yes	NRF51822	16MHz	128KB	16KB

Maxim

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
max32600mbed	Maxim ARM mbed Enabled Development Platform for MAX32600	Maxim 32	Yes	MAX326004MHz	256KB	32KB	
max32620hsp	Maxim Health Sensor Platform	Maxim 32	No	MAX326296MHz	2MB	256KB	
max32625mbed	MAX32625MBED	Maxim 32	No	MAX326296MHz	512KB	160KB	
max32625nexpaq	MAX32625NEXPAQ	Maxim 32	No	MAX326296MHz	512KB	160KB	
max32630fthr	Maxim MAX32630FTHR Application Platform	Maxim 32	No	MAX326396MHz	2MB	512KB	
maxwsnenv	Maxim Wireless Sensor Node Demonstrator	Maxim 32	Yes	MAX326104MHz	256KB	32KB	

Micromint

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lpc4330_m4	Bambino-210E	NXP LPC	Yes	LPC4330	204MHz	8MB	264KB

MikroElektronika

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
hexiwear	Hexi-wear	Freescale Kinetics	Yes	MK64FN1M0VDC12	120MHz	1MB	256KB

MultiTech

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
mts_dragonfly_f411	MTS Dragonfly	<i>ST STM32</i>	<i>Yes</i>	STM32F411RET	6100MHz	512KB	128KB
mts_mdot_f405rg	MultiTech mDot	<i>ST STM32</i>	<i>Yes</i>	STM32F405RG	6100MHz	512KB	128KB
mts_mdot_f411re	MultiTech mDot F411	<i>ST STM32</i>	<i>Yes</i>	STM32F411RET	6100MHz	512KB	128KB
xdot_1151cc	MultiTech xDot	<i>ST STM32</i>	<i>Yes</i>	STM32L151CCU	62MHz	256KB	32KB

NGX Technologies

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
blueboard_lpc11u24	NGX Technologies BlueBoard-LPC11U24	<i>NXP LPC</i>	<i>Yes</i>	LPC11U24	48MHz	32KB	8KB

NXP

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
lpc11c24	NXP LPC11C24	<i>NXP LPC</i>	<i>Yes</i>	LPC11C24	48MHz	32KB	8KB
lpc11u24	NXP mbed LPC11U24	<i>NXP LPC</i>	<i>Yes</i>	LPC11U24	48MHz	32KB	8KB
lpc11u24_30	ARM mbed LPC11U24 (+CAN)	<i>NXP LPC</i>	<i>Yes</i>	LPC11U24	48MHz	32KB	8KB
lpc11u34_42	NXP LPC11U34	<i>NXP LPC</i>	<i>Yes</i>	LPC11U34	48MHz	40KB	8KB
lpc11u37_50	NXP LPC11U37	<i>NXP LPC</i>	<i>Yes</i>	LPC11U37	48MHz	128KB	10KB
lpc11u68	LPCXpresso11U68	<i>NXP LPC</i>	<i>Yes</i>	LPC11U68	50MHz	256KB	36KB
lpc1549	NXP LPCXpresso1549	<i>NXP LPC</i>	<i>Yes</i>	LPC1549	72MHz	256KB	36KB
lpc1768	NXP mbed LPC1768	<i>NXP LPC</i>	<i>Yes</i>	LPC1768	96MHz	512KB	64KB
lpc54114	NXP LPCXpresso54114	<i>NXP LPC</i>	<i>Yes</i>	LPC54114J256BD	6400MHz	256KB	192KB
lpc546xx	NXP LPCXpresso54608	<i>NXP LPC</i>	<i>Yes</i>	LPC54608ET512	180MHz	512KB	200KB
lpc812	NXP LPC800-MAX	<i>NXP LPC</i>	<i>Yes</i>	LPC812	30MHz	16KB	4KB
lpc824	LPCXpresso824-MAX	<i>NXP LPC</i>	<i>Yes</i>	LPC824	30MHz	32KB	8KB

Nordic

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
nrf51_dk	Nordic nRF51-DK	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB
nrf51_dongle	Nordic nRF51-Dongle	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB
nrf51_mkit	Nordic nRF51822-mKIT	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	128KB	16KB
nrf52840_dk	Nordic nRF52840-DK	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52840	64MHz	1MB	256KB
nrf52_dk	Nordic nRF52-DK	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

Outrageous Circuits

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mbuino	Outrageous Circuits mBuino	<i>NXP LPC</i>	No	LPC11U24	48MHz	32KB	8KB

RedBearLab

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
redBearLab	RedBearLab nRF51822	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	256KB	16KB
redBearLabBLENano	RedBearLab BLE Nano 1.5	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	256KB	32KB

RushUp

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
cloud_jam	RushUp Cloud-JAM	<i>ST STM32</i>	<i>Yes</i>	STM32F401RET6	84MHz	512KB	96KB
cloud_jam_14	RushUp Cloud-JAM L4	<i>ST STM32</i>	<i>Yes</i>	STM32L476RGT6	80MHz	1MB	128KB

ST

ID	Name	Platform	Debug	MCU	Frequency	Flash
disco_f030r8	ST STM32F0308DISCOVERY	<i>ST STM32</i>	<i>Yes</i>	STM32F030R8T6	48MHz	64KB
disco_f051r8	ST STM32F0DISCOVERY	<i>ST STM32</i>	<i>Yes</i>	STM32F051R8T6	48MHz	64KB

Continued on

Table 8 – continued from previous page

ID	Name	Platform	Debug	MCU	Frequency	Flash
disco_f100rb	ST STM32VLDISCOVERY	ST STM32	Yes	STM32F100RBT6	24MHz	128KB
disco_f303vc	ST STM32F3DISCOVERY	ST STM32	Yes	STM32F303VCT6	72MHz	256KB
disco_f334c8	ST 32F3348DISCOVERY	ST STM32	Yes	STM32F334C8T6	72MHz	64KB
disco_f401vc	ST 32F401CDISCOVERY	ST STM32	Yes	STM32F401VCT6	84MHz	256KB
disco_f407vg	ST STM32F4DISCOVERY	ST STM32	Yes	STM32F407VGT6	168MHz	1MB
disco_f413zh	ST 32F413HDISCOVERY	ST STM32	Yes	STM32F413ZHT6	100MHz	512KB
disco_f429zi	ST 32F429IDISCOVERY	ST STM32	Yes	STM32F429ZIT6	180MHz	2MB
disco_f469ni	ST 32F469IDISCOVERY	ST STM32	Yes	STM32F469NIH6	180MHz	1MB
disco_f746ng	ST 32F746GDISCOVERY	ST STM32	Yes	STM32F746NGH6	216MHz	1MB
disco_f769ni	ST 32F769IDISCOVERY	ST STM32	Yes	STM32F769NIH6	80MHz	1MB
disco_1053c8	ST 32L0538DISCOVERY	ST STM32	Yes	STM32L053C8T6	32MHz	64KB
disco_1072cz_lrwan1	ST DISCO-L072CZ-LRWAN1	ST STM32	Yes	STM32L072CZ	32MHz	192KB
disco_1475vg_iot01a	ST DISCO-L475VG-IOT01A	ST STM32	Yes	STM32L475VGT6	80MHz	1MB
disco_1476vg	ST 32L476GDISCOVERY	ST STM32	Yes	STM32L476VGT6	80MHz	1MB
nucleo_f030r8	ST Nucleo F030R8	ST STM32	Yes	STM32F030R8T6	48MHz	64KB
nucleo_f031k6	ST Nucleo F031K6	ST STM32	Yes	STM32F031K6T6	48MHz	32KB
nucleo_f042k6	ST Nucleo F042K6	ST STM32	Yes	STM32F042K6T6	48MHz	32KB
nucleo_f070rb	ST Nucleo F070RB	ST STM32	Yes	STM32F070RBT6	48MHz	128KB
nucleo_f072rb	ST Nucleo F072RB	ST STM32	Yes	STM32F072RBT6	48MHz	128KB
nucleo_f091rc	ST Nucleo F091RC	ST STM32	Yes	STM32F091RCT6	48MHz	256KB
nucleo_f103rb	ST Nucleo F103RB	ST STM32	Yes	STM32F103RBT6	72MHz	128KB
nucleo_f207zg	ST Nucleo F207ZG	ST STM32	Yes	STM32F207ZGT6	120MHz	1MB
nucleo_f302r8	ST Nucleo F302R8	ST STM32	Yes	STM32F302R8T6	72MHz	64KB
nucleo_f303k8	ST Nucleo F303K8	ST STM32	Yes	STM32F303K8T6	72MHz	64KB
nucleo_f303re	ST Nucleo F303RE	ST STM32	Yes	STM32F303RET6	72MHz	512KB
nucleo_f303ze	ST Nucleo F303ZE	ST STM32	Yes	STM32F303ZET6	72MHz	512KB
nucleo_f334r8	ST Nucleo F334R8	ST STM32	Yes	STM32F334R8T6	72MHz	64KB
nucleo_f401re	ST Nucleo F401RE	ST STM32	Yes	STM32F401RET6	84MHz	512KB
nucleo_f410rb	ST Nucleo F410RB	ST STM32	Yes	STM32F410RBT6	100MHz	128KB
nucleo_f411re	ST Nucleo F411RE	ST STM32	Yes	STM32F411RET6	100MHz	512KB
nucleo_f412zg	ST Nucleo F412ZG	ST STM32	Yes	STM32F412ZGT6	100MHz	1MB
nucleo_f413zh	ST Nucleo F413ZH	ST STM32	Yes	STM32F413ZHT6	100MHz	512KB
nucleo_f429zi	ST Nucleo F429ZI	ST STM32	Yes	STM32F429ZIT6	180MHz	2MB
nucleo_f446re	ST Nucleo F446RE	ST STM32	Yes	STM32F446RET6	180MHz	512KB
nucleo_f446ze	ST Nucleo F446ZE	ST STM32	Yes	STM32F446ZET6	180MHz	512KB
nucleo_f746zg	ST Nucleo F746ZG	ST STM32	Yes	STM32F746ZGT6	216MHz	1MB
nucleo_f767zi	ST Nucleo F767ZI	ST STM32	Yes	STM32F767ZIT6	216MHz	2MB
nucleo_1011k4	ST Nucleo L011K4	ST STM32	Yes	STM32L011K4T6	32MHz	16KB
nucleo_1031k6	ST Nucleo L031K6	ST STM32	Yes	STM32L031K6T6	32MHz	32KB
nucleo_1053r8	ST Nucleo L053R8	ST STM32	Yes	STM32L053R8T6	32MHz	64KB
nucleo_1073rz	ST Nucleo L073RZ	ST STM32	Yes	STM32L073RZ	32MHz	192KB
nucleo_1152re	ST Nucleo L152RE	ST STM32	Yes	STM32L152RET6	32MHz	512KB
nucleo_1432kc	ST Nucleo L432KC	ST STM32	Yes	STM32L432KCU6	80MHz	256KB
nucleo_1476rg	ST Nucleo L476RG	ST STM32	Yes	STM32L476RGT6	80MHz	1MB
nucleo_1496zg	ST Nucleo L496ZG	ST STM32	Yes	STM32L496ZGT6	80MHz	1MB

SeeedStudio

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
seeedArchBLE	Seeed Arch BLE	<i>Nordic nRF51</i>	Yes	NRF51822	16MHz	128KB	16KB
seeedArchGPRS	Seeed Arch GPRS V2	<i>NXP LPC</i>	No	LPC11U37	48MHz	128KB	10KB
seeedArchLink	Seeed Arch Link	<i>Nordic nRF51</i>	Yes	NRF51822	16MHz	256KB	16KB
seeedArchMax	Seeed Arch Max	<i>ST STM32</i>	Yes	STM32F407VET6	168MHz	512KB	192KB
seeedArchPro	Seeed Arch Pro	<i>NXP LPC</i>	Yes	LPC1768	96MHz	512KB	64KB
seeedTinyBLE	Seeed Tiny BLE	<i>Nordic nRF51</i>	Yes	NRF51822	16MHz	256KB	16KB
xadow_m0	Seeed Xadow M0	<i>NXP LPC</i>	No	LPC11U35	48MHz	64KB	10KB

Semtech

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mote_1152rc	NAMote72	<i>ST STM32</i>	Yes	STM32L152RC	32MHz	256KB	32KB

Silicon Labs

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
efm32gg_st	Silicon Labs EFM32GG-STK3700 (Giant Gecko)	<i>Silicon Labs EFM32</i>	Yes	EFM32GG990	40MHz	1MB	128KB
efm32hg_st	Silicon Labs SLSTK3400A USB-enabled (Happy Gecko)	<i>Silicon Labs EFM32</i>	Yes	EFM32HG322	40MHz	64KB	8KB
efm32lg_st	Silicon Labs EFM32LG-STK3600 (Leopard Gecko)	<i>Silicon Labs EFM32</i>	Yes	EFM32LG990	25MHz	256KB	32KB
efm32pg_st	Silicon Labs SLSTK3401A (Pearl Gecko)	<i>Silicon Labs EFM32</i>	Yes	EFM32PG1B20	40MHz	256KB	32KB
efm32wg_st	Silicon Labs EFM32WG-STK3800 (Wonder Gecko)	<i>Silicon Labs EFM32</i>	Yes	EFM32WG990	25MHz	256KB	32KB
efm32zg_st	Silicon Labs EFM32ZG-STK3200 (Zero Gecko)	<i>Silicon Labs EFM32</i>	Yes	EFM32ZG222	32MHz	32KB	4KB

Smeshlink

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
xbed_lpc1768	Smeshlink LPC1768	xbed <i>NXP LPC</i>	No	LPC1768	96MHz	512KB	32KB

Solder Splash Labs

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
dipcortexm0	Solder Splash Labs DipCortex M0	<i>NXP LPC</i>	<i>Yes</i>	LPC11U24	50MHz	32KB	8KB
lpc1347	DipCortex M3	<i>NXP LPC</i>	<i>Yes</i>	LPC1347	72MHz	64KB	12KB

Switch Science

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
hrm1017	Switch Science mbed HRM1017	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	256KB	16KB
lpc1114fn28	Switch Science mbed LPC1114FN28	<i>NXP LPC</i>	<i>Yes</i>	LPC1114FN28	48MHz	32KB	4KB
ssci824	Switch Science mbed LPC824	<i>NXP LPC</i>	<i>Yes</i>	LPC824	30MHz	32KB	8KB
ty51822r3	Switch Science mbed TY51822r3	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB

Teensy

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
teensy31	Teensy 3.1 / 3.2	Teensy	<i>Yes</i>	MK20DX256	72MHz	256KB	64KB

VNG

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
vbluno51	VNG VBLUNO51	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	128KB	32KB

WIZNet

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
wizwiki_w7500	WIZwiki-W7500	WIZNet W7500	Yes	WIZNET7500	48MHz	128KB	48KB
wizwiki_w7500eco	WIZwiki-W7500ECO	WIZNet W7500	Yes	WIZ-NET7500ECO	48MHz	128KB	48KB
wizwiki_w7500p	WIZwiki-W7500P	WIZNet W7500	Yes	WIZ-NET7500P	48MHz	128KB	48KB

u-blox

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
mbed_connect_odin	u-blox Connect Cloud	ST STM32	Yes	STM32F439ZI	168MHz	2MB	256KB
mtb_ublox_odin	u-blox ODIN-W2	ST STM32	Yes	STM32F439ZI	168MHz	2MB	256KB
ublox_c030_n21	u-blox C030-N211 IoT Starter Kit	ST STM32	Yes	STM32F437VG	180MHz	1MB	256KB
ublox_c030_u20	u-blox C030-U201 IoT Starter Kit	ST STM32	Yes	STM32F437VG	180MHz	1MB	256KB
ublox_evk_nina	u-blox EVK-NINA-B1	Nordic nRF52	Yes	NRF52832	64MHz	512KB	64KB
ublox_evk_odin	u-blox EVK-ODIN-W2	ST STM32	Yes	STM32F439ZI	168MHz	2MB	256KB
ubloxc027	u-blox C027	NXP LPC	Yes	LPC1768	96MHz	512KB	64KB

y5 design

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
lpc11u35_y5_mbug	y5 LPC11U35 mbug	NXP LPC	Yes	LPC11U35	48MHz	64KB	10KB
nrf51822_y5_mbug	y5 nRF51822 mbug	Nordic nRF51	Yes	NRF51822	16MHz	256KB	16KB

1.11.11 Pumbaa

framework = pumbaa

Pumbaa is Python on top of Simba. The implementation is a port of MicroPython, designed for embedded devices with limited amount of RAM and code memory.

For more detailed information please visit [vendor site](#).

Contents

- Examples

- *Platforms*
- *Boards*

Examples

- Pumbaa for Espressif 32

Platforms

Name	Description
Espressif 32	Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

MakerAsia

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
nano32	MakerAsia Nano32	Espressif 32	No	ESP32	240MHz	4MB	320KB

1.11.12 Simba

`framework = simba`

Simba is an RTOS and build framework. It aims to make embedded programming easy and portable.

For more detailed information please visit [vendor site](#).

Contents

- *Debugging*
- *Examples*
- *Platforms*
- *Boards*

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

External Debug Tools

Banks listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
due	Arduino Due (Programming Port)	<i>Atmel SAM</i>	<i>Black Magic Probe, J-LINK</i>	AT91SAM3X8E	4MHz	512KB	32KB
dueUSB	Arduino Due (USB Native Port)	<i>Atmel SAM</i>	<i>Black Magic Probe, J-LINK</i>	AT91SAM3X8E	4MHz	512KB	32KB

Examples

- Simba for Atmel AVR
- Simba for Atmel SAM
- Simba for Espressif 32
- Simba for Espressif 8266

Platforms

Name	Description
<i>Atmel AVR</i>	Atmel AVR 8- and 32-bit MCUs deliver a unique combination of performance, power efficiency and design flexibility. Optimized to speed time to market-and easily adapt to new ones-they are based on the industry's most code-efficient architecture for C and assembly programming.
<i>Atmel SAM</i>	Atmel SMART offers Flash- based ARM products based on the ARM Cortex-M0+, Cortex-M3 and Cortex-M4 architectures, ranging from 8KB to 2MB of Flash including a rich peripheral and feature mix.
<i>Espressif 32</i>	Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.
<i>Espressif 8266</i>	Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

Adafruit

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
huzzah	Adafruit HUZZAH ESP8266	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

Arduino

ID	Name	Plat- form	De- bug	MCU	Fre- quency	Flash	RAM
due	Arduino Due (Programming Port)	<i>Atmel SAM</i>	<i>Yes</i>	AT91SAM3X8E	48MHz	512KB	32KB
dueUSB	Arduino Due (USB Native Port)	<i>Atmel SAM</i>	<i>Yes</i>	AT91SAM3X8E	48MHz	512KB	32KB
megaatmega2560	Arduino Mega or Mega 2560 ATmega2560 (Mega 2560)	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	248KB	8KB
nanoatmega328	Arduino Nano ATmega328	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	30KB	2KB
uno	Arduino Uno	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB

Espressif

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
esp01	Espressif Generic ESP8266 ESP-01 512k	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB
esp12e	Espressif ESP8266 ESP-12E	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
esp_wroom_02	ESP-WROOM-02	<i>Espressif 8266</i>	No	ESP8266	80MHz	2MB	80KB

MakerAsia

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
nano32	MakerAsia Nano32	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

NodeMCU

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
nodemcu	NodeMCU 0.9 (ESP-12 Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
nodemcuv2	NodeMCU 1.0 (ESP-12E Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

SeeedStudio

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
seeeduino	Seeeduino	<i>Atmel AVR</i>	No	ATMEGA328P	16MHz	31.50KB	2KB

1.11.13 SPL

`framework = spl`

The ST Standard Peripheral Library provides a set of functions for handling the peripherals on the STM32 Cortex-M3 family. The idea is to save the user (the new user, in particular) having to deal directly with the registers.

For more detailed information please visit [vendor site](#).

Contents

- [Examples](#)
- [Debugging](#)
- [Examples](#)

- *Platforms*
- *Boards*

Examples

All project examples are located in PlatformIO repository [Examples for SPL framework](#).

- [Blink](#)

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging [Tools & Debug Probes](#) using [debug_tool](#) options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Bands listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
cloud_jam	RushUp Cloud-JAM	ST STM32	ST-LINK (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32F401RE	48MHz	512KB	96KB
disco_f303	STM32F3DISCO	ST STM32	ST-LINK (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32F303VCT6	48MHz	256KB	48KB
disco_f407	STM32F4DISCO	ST STM32	ST-LINK (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32F407VCT6	48MHz	1MB	128KB
disco_l152	STM32LDISCO	ST STM32	ST-LINK (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32L152RE	48MHz	128KB	16KB
nucleo_f401re	Nucleo F401RE	ST STM32	ST-LINK (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32F401RE	48MHz	512KB	96KB

External Debug Tools

Boards listed below are compatible with [PIO Unified Debugger](#) but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
1bitsy_stm32f415rgt	1BitS ^q	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F415RGT	168MHz	1MB	128KB
armstrap_eagle1024	1024strap Eagle 1024	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F417VGT	168MHz	1MB	192KB
armstrap_eagle2048	2048strap Eagle 2048	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F427VIT	168MHz	1.99MB	256KB
armstrap_eagle512	512strap Eagle 512	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F407VHT	168MHz	512KB	192KB

Examples

- SPL for ST STM32

Platforms

Name	Description
ST STM32	The STM32 family of 32-bit Flash MCUs based on the ARM Cortex-M processor is designed to offer new degrees of freedom to MCU users. It offers a 32-bit product range that combines very high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

1BitSquared

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
1bitsy_stm32f415rgt	1Bitsy	ST STM32	Yes	STM32F415RGT	168MHz	1MB	128KB

Armstrap

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
armstrap_eagle1024	Armstrap Eagle 1024	ST STM32	Yes	STM32F417VGT6	168MHz	1MB	192KB
armstrap_eagle2048	Armstrap Eagle 2048	ST STM32	Yes	STM32F427VIT6	168MHz	1.99MB	256KB
armstrap_eagle512	Armstrap Eagle 512	ST STM32	Yes	STM32F407VET6	168MHz	512KB	192KB

RushUp

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
cloud_jam	RushUp Cloud-JAM	ST STM32	Yes	STM32F401RET6	84MHz	512KB	96KB

ST

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
disco_f303vc	ST STM32F3DISCOVERY	ST STM32	Yes	STM32F303VCT6	72MHz	256KB	48KB
disco_f407vg	ST STM32F4DISCOVERY	ST STM32	Yes	STM32F407VGT6	168MHz	1MB	128KB
disco_l152rb	ST STM32LDISCOVERY	ST STM32	Yes	STM32L152RBT6	32MHz	128KB	16KB
nucleo_f401re	ST Nucleo F401RE	ST STM32	Yes	STM32F401RET6	84MHz	512KB	96KB

1.11.14 STM32Cube

framework = stm32cube

STM32Cube embedded software libraries, including: The HAL hardware abstraction layer, enabling portability between different STM32 devices via standardized API calls; The Low-Layer (LL) APIs, a light-weight, optimized, expert oriented set of APIs designed for both performance and runtime efficiency.

For more detailed information please visit [vendor site](#).

Contents

- [Tutorials](#)
- [Debugging](#)
- [Examples](#)

- *Platforms*
- *Boards*

Tutorials

- *STM32Cube HAL and Nucleo-F401RE: debugging and unit testing*

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- *Debug Tools*
 - *On-Board Debug Tools*
 - *External Debug Tools*

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

On-Board Debug Tools

Bands listed below have on-board debug tool and **ARE READY** for debugging! You do not need to use/buy external debug tool.

ID	Name	Platform	Debug
b96b_f446ve	96Boards B96B-F446VE	ST STM32	ST-LINK (default, on-board)
cloud_jam	RushUp Cloud-JAM	ST STM32	ST-LINK (default, on-board)
cloud_jam_14	RushUp Cloud-JAM L4	ST STM32	ST-LINK (default, on-board)
disco_f030r8	ST STM32F0308DISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f051r8	ST STM32F0DISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f100rb	ST STM32VLDISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f303vc	ST STM32F3DISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f334c8	ST 32F3348DISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f401vc	ST 32F401CDISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f407vg	ST STM32F4DISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f411ve	ST 32F411EDISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f413zh	ST 32F413HDISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f429zi	ST 32F429IDISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f469ni	ST 32F469IDISCOVERY	ST STM32	ST-LINK (default, on-board)

Table 9 – continued from previous page

ID	Name	Platform	Debug
disco_f746ng	ST 32F746GDISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_f769ni	ST 32F769IDISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_l053c8	ST 32L0538DISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_l072cz_lrwan1	ST DISCO-L072CZ-LRWAN1	ST STM32	ST-LINK (default, on-board)
disco_l152rb	ST STM32LDISCOVERY	ST STM32	ST-LINK (default, on-board)
disco_l475vg_iot01a	ST DISCO-L475VG-IOT01A	ST STM32	ST-LINK (default, on-board)
disco_l476vg	ST 32L476GDISCOVERY	ST STM32	ST-LINK (default, on-board)
eval_l073z	ST STM32L073Z-EVAL	ST STM32	ST-LINK (default, on-board)
mbed_connect_odin	Mbed Connect Cloud	ST STM32	CMSIS-DAP (on-board), Baud
mxchip_az3166	Microsoft Azure IoT Development Kit (MXChip AZ3166)	ST STM32	ST-LINK (default, on-board)
nucleo_f030r8	ST Nucleo F030R8	ST STM32	ST-LINK (default, on-board)
nucleo_f031k6	ST Nucleo F031K6	ST STM32	ST-LINK (default, on-board)
nucleo_f042k6	ST Nucleo F042K6	ST STM32	ST-LINK (default, on-board)
nucleo_f070rb	ST Nucleo F070RB	ST STM32	ST-LINK (default, on-board)
nucleo_f072rb	ST Nucleo F072RB	ST STM32	ST-LINK (default, on-board)
nucleo_f091rc	ST Nucleo F091RC	ST STM32	ST-LINK (default, on-board)
nucleo_f103rb	ST Nucleo F103RB	ST STM32	ST-LINK (default, on-board)
nucleo_f207zg	ST Nucleo F207ZG	ST STM32	ST-LINK (default, on-board)
nucleo_f302r8	ST Nucleo F302R8	ST STM32	ST-LINK (default, on-board)
nucleo_f303k8	ST Nucleo F303K8	ST STM32	ST-LINK (default, on-board)
nucleo_f303re	ST Nucleo F303RE	ST STM32	ST-LINK (default, on-board)
nucleo_f303ze	ST Nucleo F303ZE	ST STM32	ST-LINK (default, on-board)
nucleo_f334r8	ST Nucleo F334R8	ST STM32	ST-LINK (default, on-board)
nucleo_f401re	ST Nucleo F401RE	ST STM32	ST-LINK (default, on-board)
nucleo_f410rb	ST Nucleo F410RB	ST STM32	ST-LINK (default, on-board)
nucleo_f411re	ST Nucleo F411RE	ST STM32	ST-LINK (default, on-board)
nucleo_f412zg	ST Nucleo F412ZG	ST STM32	ST-LINK (default, on-board)
nucleo_f413zh	ST Nucleo F413ZH	ST STM32	ST-LINK (default, on-board)
nucleo_f429zi	ST Nucleo F429ZI	ST STM32	ST-LINK (default, on-board)
nucleo_f446re	ST Nucleo F446RE	ST STM32	ST-LINK (default, on-board)
nucleo_f446ze	ST Nucleo F446ZE	ST STM32	ST-LINK (default, on-board)
nucleo_f746zg	ST Nucleo F746ZG	ST STM32	ST-LINK (default, on-board)
nucleo_f767zi	ST Nucleo F767ZI	ST STM32	ST-LINK (default, on-board)
nucleo_l011k4	ST Nucleo L011K4	ST STM32	ST-LINK (default, on-board)
nucleo_l031k6	ST Nucleo L031K6	ST STM32	ST-LINK (default, on-board)
nucleo_l053r8	ST Nucleo L053R8	ST STM32	ST-LINK (default, on-board)
nucleo_l073rz	ST Nucleo L073RZ	ST STM32	ST-LINK (default, on-board)
nucleo_l152re	ST Nucleo L152RE	ST STM32	ST-LINK (default, on-board)
nucleo_l432kc	ST Nucleo L432KC	ST STM32	ST-LINK (default, on-board)
nucleo_l476rg	ST Nucleo L476RG	ST STM32	ST-LINK (default, on-board)
nucleo_l496zg	ST Nucleo L496ZG	ST STM32	ST-LINK (default, on-board)
seeedArchMax	Seeed Arch Max	ST STM32	ST-LINK (default, on-board)

External Debug Tools

Banks listed below are compatible with [PIO Unified Debugger](#) but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
1bitsy_stm32f110rgt		ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F411REMHz	1MHz	1MB	128KB
armstrap_eagle1024	Eagle 1024	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F417VCTMHz	1MHz	1MB	192KB
armstrap_eagle2048	Eagle 2048	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F427VCTMHz	1.99MHz	256KB	
armstrap_eagle512	Eagle 512	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F407VCTMHz	512KB	192KB	
bluepill_f103c8	BluePill F103C8	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103C8MHz	64KB	20KB	
elmo_f411re	Espotel LoRa Mod-ule	ST STM32	Black Magic Probe, J-LINK, ST-LINK (default)	STM32F411REMHz	512KB	128KB	
genericSTM32F03C8	(20k RAM. 64k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103C8MHz	64KB	20KB	
genericSTM32F03CB	(20k RAM. 128k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	128KB	20KB	
genericSTM32F03R8	(20k RAM. 64 Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	64KB	20KB	
genericSTM32F03RB	(20k RAM. 128k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	128KB	20KB	
genericSTM32F03RC	(48k RAM. 256k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	256KB	48KB	
genericSTM32F03RE	(64k RAM. 512k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	512KB	64KB	
genericSTM32F03VC	(48k RAM. 256k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103VCTMHz	256KB	48KB	
genericSTM32F03VE	(64k RAM. 512k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103VCTMHz	512KB	64KB	
genericSTM32F407VE	(192k RAM. 512k Flash)	ST STM32	ST-LINK	STM32F407VCTMHz	502.23MHz	28KB	
maple	Maple	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	108KB	17KB	
maple_mini_bootloader	Maple Mini Boot-loader 2.0	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	120KB	20KB	
maple_mini_original	Maple Mini Original	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	108KB	17KB	
mote_1152rc	NAMote72	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32L152RMHz	256KB	32KB	
mtb_ublox_odinblox	ODIN-W2	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F439ZV8MHz	2MB	256KB	
mts_dragonfly	Dragonfly	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F411REMHz	512KB	128KB	
mts_mdot_f405	MultiTech mDot	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F411REMHz	512KB	128KB	
mts_mdot_f411	MultiTech mDot F411	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F411REMHz	512KB	128KB	
ublox_c030_frameworks	ublox C030-N211 IoT Starter Kit	ST STM32	Black Magic Probe, CMSIS-DAP, J-LINK, ST-LINK	STM32F437VQMHz	1MB	256KB	429
ublox_c030_u20	ublox C030-U201 IoT Starter Kit	ST STM32	Black Magic Probe, CMSIS-DAP, J-LINK	STM32F437VQMHz	1MB	256KB	

Examples

- STM32Cube for ST STM32

Platforms

Name	Description
ST STM32	The STM32 family of 32-bit Flash MCUs based on the ARM Cortex-M processor is designed to offer new degrees of freedom to MCU users. It offers a 32-bit product range that combines very high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.

Boards

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

1BitSquared

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
1bitsy_stm32f415rgt	1Bitsy	ST STM32	Yes	STM32F415RGT	168MHz	1MB	128KB

96Boards

ID	Name	Plat- form	De- bug	MCU	Fre- quency	Flash	RAM
b96b_f446ve	96Boards B96B-F446VE	ST STM32	Yes	STM32F446VET6	168MHz	512KB	128KB

Armstrap

ID	Name	Plat- form	De- bug	MCU	Fre- quency	Flash	RAM
armstrap_eagle10	Armstrap Eagle 1024	ST STM32	Yes	STM32F417VGT6	168MHz	1MB	192KB
armstrap_eagle20	Armstrap Eagle 2048	ST STM32	Yes	STM32F427VIT6	168MHz	1.99MB	256KB
armstrap_eagle51	Armstrap Eagle 512	ST STM32	Yes	STM32F407VET6	168MHz	512KB	192KB

Espotel

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
elmo_f411re	Espotel LoRa Mod-ule	ST STM32	Yes	STM32F411RET6	100MHz	512KB	128KB

Generic

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
bluepill_f103c8	BluePill F103C8	ST STM32	Yes	STM32F103C8T6MHz	8MHz	64KB	20KB
genericSTM32F103C8	STM32F103C8 (20k RAM. 64k Flash)	ST STM32	Yes	STM32F103C8T6MHz	8MHz	64KB	20KB
genericSTM32F103CB	STM32F103CB (20k RAM. 128k Flash)	ST STM32	Yes	STM32F103CBT6MHz	128MHz	128KB	20KB
genericSTM32F103R8	STM32F103R8 (20k RAM. 64 Flash)	ST STM32	Yes	STM32F103R8T6MHz	8MHz	64KB	20KB
genericSTM32F103RB	STM32F103RB (20k RAM. 128k Flash)	ST STM32	Yes	STM32F103RBT6MHz	128MHz	128KB	20KB
genericSTM32F103RC	STM32F103RC (48k RAM. 256k Flash)	ST STM32	Yes	STM32F103RCT6MHz	128MHz	256KB	48KB
genericSTM32F103RE	STM32F103RE (64k RAM. 512k Flash)	ST STM32	Yes	STM32F103RET6MHz	128MHz	512KB	64KB
genericSTM32F103VC	STM32F103VC (48k RAM. 256k Flash)	ST STM32	Yes	STM32F103VCT6MHz	128MHz	256KB	48KB
genericSTM32F103VE	STM32F103VE (64k RAM. 512k Flash)	ST STM32	Yes	STM32F103VET6MHz	128MHz	512KB	64KB
genericSTM32F407VE	STM32F407VE (192k RAM. 512k Flash)	ST STM32	Yes	STM32F407VET6MHz	128MHz	502.23KB	128KB

LeafLabs

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
maple	Maple	ST STM32	Yes	STM32F103RBT6/2MHz	108KB	17KB	
maple_mini_b20	Maple Mini Boot-loader 2.0	ST STM32	Yes	STM32F103CBT6/2MHz	120KB	20KB	
maple_mini_origin	Maple Mini Original	ST STM32	Yes	STM32F103CBT6/2MHz	108KB	17KB	

MXChip

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mxchip_az3	Microsoft Azure IoT Development Kit (MXChip AZ3166)	ST STM32	Yes	STM32F412ZG	100MHz	1MB	256KB

MultiTech

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mts_dragonfly_f411	MTS Dragonfly	ST STM32	Yes	STM32F411RET	6100MHz	512KB	128KB
mts_mdot_f405rg	MultiTech mDot	ST STM32	Yes	STM32F411RET	6100MHz	512KB	128KB
mts_mdot_f411re	MultiTech mDot F411	ST STM32	Yes	STM32F411RET	6100MHz	512KB	128KB
xdot_l1151cc	MultiTech xDot	ST STM32	Yes	STM32L151CCU	62MHz	256KB	32KB

RushUp

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
cloud_jam	RushUp Cloud-JAM	ST STM32	Yes	STM32F401RET	84MHz	512KB	96KB
cloud_jam_l4	RushUp Cloud-JAM L4	ST STM32	Yes	STM32L476RGT	80MHz	1MB	128KB

ST

ID	Name	Platform	Debug	MCU	Frequency	Flash
disco_f030r8	ST STM32F0308DISCOVERY	ST STM32	Yes	STM32F030R8T6	48MHz	64KB
disco_f051r8	ST STM32F0DISCOVERY	ST STM32	Yes	STM32F051R8T6	48MHz	64KB
disco_f100rb	ST STM32VLDISCOVERY	ST STM32	Yes	STM32F100RBT6	24MHz	128KB
disco_f303vc	ST STM32F3DISCOVERY	ST STM32	Yes	STM32F303VCT6	72MHz	256KB
disco_f334c8	ST 32F3348DISCOVERY	ST STM32	Yes	STM32F334C8T6	72MHz	64KB
disco_f401vc	ST 32F401CDISCOVERY	ST STM32	Yes	STM32F401VCT6	84MHz	256KB
disco_f407vg	ST STM32F4DISCOVERY	ST STM32	Yes	STM32F407VGT6	168MHz	1MB
disco_f411ve	ST 32F411EDISCOVERY	ST STM32	Yes	STM32F411VET6	100MHz	512KB
disco_f413zh	ST 32F413HDISCOVERY	ST STM32	Yes	STM32F413ZHT6	100MHz	512KB
disco_f429zi	ST 32F429IDISCOVERY	ST STM32	Yes	STM32F429ZIT6	180MHz	2MB
disco_f469ni	ST 32F469IDISCOVERY	ST STM32	Yes	STM32F469NIH6	180MHz	1MB
disco_f746ng	ST 32F746GDISCOVERY	ST STM32	Yes	STM32F746NGH6	216MHz	1MB
disco_f769ni	ST 32F769IDISCOVERY	ST STM32	Yes	STM32F769NIH6	80MHz	1MB

Continued on

Table 10 – continued from previous page

ID	Name	Platform	Debug	MCU	Frequency	Flash
disco_1053c8	ST 32L0538DISCOVERY	ST STM32	Yes	STM32L053C8T6	32MHz	64KB
disco_1072cz_lrwan1	ST DISCO-L072CZ-LRWAN1	ST STM32	Yes	STM32L072CZ	32MHz	192KB
disco_1152rb	ST STM32LDISCOVERY	ST STM32	Yes	STM32L152RBT6	32MHz	128KB
disco_1475vg_iot01a	ST DISCO-L475VG-IOT01A	ST STM32	Yes	STM32L475VGT6	80MHz	1MB
disco_1476vg	ST 32L476GDISCOVERY	ST STM32	Yes	STM32L476VGT6	80MHz	1MB
eval_1073z	ST STM32L073Z-EVAL	ST STM32	Yes	STM32L073VZT6	32MHz	192KB
nucleo_f030r8	ST Nucleo F030R8	ST STM32	Yes	STM32F030R8T6	48MHz	64KB
nucleo_f031k6	ST Nucleo F031K6	ST STM32	Yes	STM32F031K6T6	48MHz	32KB
nucleo_f042k6	ST Nucleo F042K6	ST STM32	Yes	STM32F042K6T6	48MHz	32KB
nucleo_f070rb	ST Nucleo F070RB	ST STM32	Yes	STM32F070RBT6	48MHz	128KB
nucleo_f072rb	ST Nucleo F072RB	ST STM32	Yes	STM32F072RBT6	48MHz	128KB
nucleo_f091rc	ST Nucleo F091RC	ST STM32	Yes	STM32F091RCT6	48MHz	256KB
nucleo_f103rb	ST Nucleo F103RB	ST STM32	Yes	STM32F103RBT6	72MHz	128KB
nucleo_f207zg	ST Nucleo F207ZG	ST STM32	Yes	STM32F207ZGT6	120MHz	1MB
nucleo_f302r8	ST Nucleo F302R8	ST STM32	Yes	STM32F302R8T6	72MHz	64KB
nucleo_f303k8	ST Nucleo F303K8	ST STM32	Yes	STM32F303K8T6	72MHz	64KB
nucleo_f303re	ST Nucleo F303RE	ST STM32	Yes	STM32F303RET6	72MHz	512KB
nucleo_f303ze	ST Nucleo F303ZE	ST STM32	Yes	STM32F303ZET6	72MHz	512KB
nucleo_f334r8	ST Nucleo F334R8	ST STM32	Yes	STM32F334R8T6	72MHz	64KB
nucleo_f401re	ST Nucleo F401RE	ST STM32	Yes	STM32F401RET6	84MHz	512KB
nucleo_f410rb	ST Nucleo F410RB	ST STM32	Yes	STM32F410RBT6	100MHz	128KB
nucleo_f411re	ST Nucleo F411RE	ST STM32	Yes	STM32F411RET6	100MHz	512KB
nucleo_f412zg	ST Nucleo F412ZG	ST STM32	Yes	STM32F412ZGT6	100MHz	1MB
nucleo_f413zh	ST Nucleo F413ZH	ST STM32	Yes	STM32F413ZHT6	100MHz	512KB
nucleo_f429zi	ST Nucleo F429ZI	ST STM32	Yes	STM32F429ZIT6	180MHz	2MB
nucleo_f446re	ST Nucleo F446RE	ST STM32	Yes	STM32F446RET6	180MHz	512KB
nucleo_f446ze	ST Nucleo F446ZE	ST STM32	Yes	STM32F446ZET6	180MHz	512KB
nucleo_f746zg	ST Nucleo F746ZG	ST STM32	Yes	STM32F746ZGT6	216MHz	1MB
nucleo_f767zi	ST Nucleo F767ZI	ST STM32	Yes	STM32F767ZIT6	216MHz	2MB
nucleo_1011k4	ST Nucleo L011K4	ST STM32	Yes	STM32L011K4T6	32MHz	16KB
nucleo_1031k6	ST Nucleo L031K6	ST STM32	Yes	STM32L031K6T6	32MHz	32KB
nucleo_1053r8	ST Nucleo L053R8	ST STM32	Yes	STM32L053R8T6	32MHz	64KB
nucleo_1073rz	ST Nucleo L073RZ	ST STM32	Yes	STM32L073RZ	32MHz	192KB
nucleo_1152re	ST Nucleo L152RE	ST STM32	Yes	STM32L152RET6	32MHz	512KB
nucleo_1432kc	ST Nucleo L432KC	ST STM32	Yes	STM32L432KCU6	80MHz	256KB
nucleo_1476rg	ST Nucleo L476RG	ST STM32	Yes	STM32L476RGT6	80MHz	1MB
nucleo_1496zg	ST Nucleo L496ZG	ST STM32	Yes	STM32L496ZGT6	80MHz	1MB

SeeedStudio

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
seeeedArchMax	Seeed Max	Arch	ST STM32	Yes	STM32F407VET6	168MHz	512KB

Semtech

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mote_1152rc	NAMote72	ST STM32	Yes	STM32L152RC	32MHz	256KB	32KB

u-blox

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
mbed_connect_odin	Mbed Connect Cloud	ST STM32	Yes	STM32F439ZIY	668MHz	2MB	256KB
mtb_ublox_odin	u-blox ODIN-W2	ST STM32	Yes	STM32F439ZIY	668MHz	2MB	256KB
ublox_c030_n21	u-blox C030-N211 IoT Starter Kit	ST STM32	Yes	STM32F437VG	180MHz	1MB	256KB
ublox_c030_u20	u-blox C030-U201 IoT Starter Kit	ST STM32	Yes	STM32F437VG	180MHz	1MB	256KB
ublox_evk_odin	u-blox EVK-ODIN-W2	ST STM32	Yes	STM32F439ZIY	668MHz	2MB	256KB

1.11.15 Tizen RT

framework = tizenrt

Tizen RT is a lightweight RTOS-based platform to support low-end IoT devices

For more detailed information please visit [vendor site](#).

Contents

- [Debugging](#)
- [Examples](#)
- [Platforms](#)
- [Boards](#)

Debugging

PIO Unified Debugger - “1-click” solution for debugging with a zero configuration.

- [Debug Tools](#)
 - [External Debug Tools](#)

Debug Tools

Supported debugging tools are listed in “Debug” column. For more detailed information, please scroll table by horizontal. You can switch between debugging *Tools & Debug Probes* using *debug_tool* options.

Warning: You will need to install debug tool drivers depending on your system. Please click on compatible debug tool below for the further instructions.

External Debug Tools

Banks listed below are compatible with *PIO Unified Debugger* but **DEPEND ON** external debug tool. See “Debug” column for compatible debug tools.

ID	Name	Platform	Debug	MCU	Fre-quency	Flash	RAM
artik_053	Samsung AR-TIK053	Samsung AR-TIK	FTDI Chip (default)	S5JT200	320MHz	8MB	1.25MB

Examples

- Tizen RT for Samsung ARTIK

Platforms

Name	Description
Samsung ARTIK	The Samsung ARTIK Smart IoT platform brings hardware modules and cloud services together, with built-in security and an ecosystem of tools and partners to speed up your time-to-market.

Boards

Note:

- You can list pre-configured boards by *platformio boards* command or PlatformIO Boards Explorer
- For more detailed board information please scroll tables below by horizontal.

Samsung

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
artik_053	Samsung AR-TIK053	Samsung AR-TIK	Yes	S5JT200	320MHz	8MB	1.25MB

1.11.16 WiringPi

framework = wiringpi

WiringPi is a GPIO access library written in C for the BCM2835 used in the Raspberry Pi. It's designed to be familiar to people who have used the Arduino "wiring" system.

For more detailed information please visit [vendor site](#).

Contents

- [Examples](#)
- [Platforms](#)
- [Boards](#)

Examples

- [WiringPi for Linux ARM](#)

Platforms

Name	Description
Linux ARM	Linux ARM is a Unix-like and mostly POSIX-compliant computer operating system (OS) assembled under the model of free and open-source software development and distribution. Using host OS (Mac OS X, Linux ARM) you can build native application for Linux ARM platform.

Boards

Note:

- You can list pre-configured boards by [platformio boards](#) command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

Raspberry Pi

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
raspberrypi_1b	Raspberry Pi 1 Model B	Linux ARM	No	BCM2835	700MHz	512MB	512MB
raspberrypi_2b	Raspberry Pi 2 Model B	Linux ARM	No	BCM2836	900MHz	1GB	1GB
raspberrypi_3b	Raspberry Pi 3 Model B	Linux ARM	No	BCM2837	1200MHz	1GB	1GB
raspberrypi_zero	Raspberry Pi Zero	Linux ARM	No	BCM2835	1000MHz	512MB	512MB

1.12 Embedded Boards

Rapid Embedded Development, Continuous and IDE integration in a few steps with PlatformIO thanks to built-in project generator for the most popular embedded boards and IDE.

Note:

- You can list pre-configured boards by `platformio boards` command or PlatformIO Boards Explorer
 - For more detailed board information please scroll tables below by horizontal.
-

Vendors

- [*IBitSquared*](#)
- [*4D Systems*](#)
- [*4DSystems*](#)
- [*96Boards*](#)
- [*Adafruit*](#)
- [*Aiyarafun*](#)
- [*Alorium Technology*](#)
- [*Anarduino*](#)
- [*AppNearMe*](#)
- [*April Brother*](#)
- [*Arduboy*](#)
- [*Arduino*](#)
- [*Armstrap*](#)
- [*Atmel*](#)
- [*BBC*](#)
- [*BQ*](#)
- [*BitWizard*](#)
- [*BluzDK*](#)
- [*CQ Publishing*](#)
- [*Controllino*](#)
- [*DFRobot*](#)
- [*DOIT*](#)
- [*Delta*](#)
- [*DigiStump*](#)
- [*Digilent*](#)
- [*Digistump*](#)

- *Doit*
- *Dongsen Technology*
- *Dwengo*
- *DycodeX*
- *ESP32vn*
- *ESPert*
- *ESPino*
- *Electronic SweetPeas*
- *Elektor*
- *Elektor Labs*
- *Embedded Artists*
- *Engduino*
- *EnviroDIY*
- *Espotel*
- *Espressif*
- *FPGAwars*
- *Freescale*
- *Fubarino*
- *GHI Electronics*
- *Generic*
- *Hardkernel*
- *Heltec*
- *Heltec Automation*
- *Hornbill*
- *Infineon*
- *Intel*
- *IntoRobot*
- *JKSoft*
- *Lattice*
- *LeafLabs*
- *LightUp*
- *Linino*
- *LowPowerLab*
- *M5Stack*
- *MH-ET Live*

- *MXChip*
- *Macchina*
- *MakerAsia*
- *Maxim*
- *Mcudude*
- *MediaTek Labs*
- *Microchip*
- *Microduino*
- *Micromint*
- *MikroElektronika*
- *MultiTech*
- *NGX Technologies*
- *NXP*
- *NodeMCU*
- *Noduino*
- *Nordic*
- *OLIMEX*
- *OSHChip*
- *Olimex*
- *Onehorse*
- *OpenBCI*
- *OpenEnergyMonitor*
- *Outrageous Circuits*
- *PONTECH*
- *PanStamp*
- *Pinoccio*
- *Pololu Corporation*
- *Pontech*
- *Punch Through*
- *Quirkbot*
- *RFduino*
- *Raspberry Pi*
- *RedBearLab*
- *RepRap*
- *RoboticsBrno*

- *RushUp*
- *SODAQ*
- *ST*
- *SainSmart*
- *Samsung*
- *Sanguino*
- *SeeedStudio*
- *Semtech*
- *SiFive*
- *Silicon Labs*
- *Smeshlink*
- *Solder Splash Labs*
- *SparkFun*
- *SparkFun Electronics*
- *SpellFoundry*
- *SweetPea*
- *Switch Science*
- *TI*
- *TTGO*
- *Taida Century*
- *Teensy*
- *ThaiEasyElec*
- *The Things Network*
- *TinyCircuits*
- *UBW32*
- *VNG*
- *WEMOS*
- *WIZNet*
- *Waveshare*
- *Wicked Device*
- *Widora*
- *Xilinx*
- *XinaBox*
- *chipKIT*
- *element14*

- *makerlab.mx*
- *ng-beacon*
- *nicai-systems*
- *u-blox*
- *ubIQio*
- *y5 design*

1.12.1 1BitSquared

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
1bitsy_stm32f415rgt	1Bitsy	ST STM32	Yes	STM32F415RGT	168MHz	1MB	128KB

1.12.2 4D Systems

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
gen4iod	4D Systems gen4 IoD Range	Espressif 8266	No	ESP8266	80MHz	512KB	80KB

1.12.3 4DSystems

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
picadillo_35t	4DSystems PI-Cadillo 35T	Microchip PIC32	No	32MX795F512L	80MHz	508KB	128KB

1.12.4 96Boards

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
b96b_f446ve	96Boards B96B-F446VE	ST STM32	Yes	STM32F446VET6	168MHz	512KB	128KB

1.12.5 Adafruit

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
adafruit_circuitplay	Adafruit Circuit Playground Express	Atmel SAM	Yes	SAMD21G188MHz	256KB	32KB	
adafruit_feather_m0	Adafruit Feather M0	Atmel SAM	Yes	SAMD21G188MHz	256KB	32KB	
adafruit_feather_m0	Adafruit Feather M0 Express	Atmel SAM	Yes	SAMD21G188MHz	256KB	32KB	
adafruit_feather_m4	Adafruit Feather M4 (SAMD51)	Atmel SAM	Yes	SAMD51J19120MHz	496KB	192KB	
adafruit_gemma_m0	Adafruit Gemma M0	Atmel SAM	Yes	SAMD21E188MHz	256KB	32KB	
adafruit_itsybitsy_m0	Adafruit ItsyBitsy M0	Atmel SAM	Yes	SAMD21G188MHz	256KB	32KB	
adafruit_itsybitsy_m0	Adafruit ItsyBitsy M4 (SAMD51)	Atmel SAM	Yes	SAMD51J19120MHz	496KB	192KB	
adafruit_metro_m0	Adafruit Metro M0 Express	Atmel SAM	Yes	SAMD21G188MHz	256KB	32KB	
adafruit_metro_m4	Adafruit Metro M4 (SAMD51)	Atmel SAM	Yes	SAMD51J19120MHz	496KB	192KB	
adafruit_pirkey	Adafruit pIRkey	Atmel SAM	Yes	SAMD21E188MHz	256KB	32KB	
adafruit_trinket_m0	Adafruit Trinket M0	Atmel SAM	Yes	SAMD21E188MHz	256KB	32KB	
bluefruitmicro	Adafruit Bluefruit Micro	Atmel AVR	No	AT-MEGA32U4	8MHz	28KB	2.50KB
feather328p	Adafruit Feather 328P	Atmel AVR	No	AT-MEGA328P	8MHz	31.50KB	2KB
feather32u4	Adafruit Feather 32u4	Atmel AVR	No	AT-MEGA32U4	8MHz	28KB	2.50KB
featheresp32	Adafruit ESP32 Feather	Espres-sif 32	Yes	ESP32	240MHz	4MB	320KB
flora8	Adafruit Flora	Atmel AVR	No	AT-MEGA32U4	8MHz	28KB	2.50KB
gemma	Adafruit Gemma	Atmel AVR	No	AT-TINY85	8MHz	8KB	512B
huzzah	Adafruit HUZZAH ESP8266	Espres-sif 8266	No	ESP8266	80MHz	4MB	80KB
itsybitsy32u4_3V	Adafruit ItsyBitsy 3V/8MHz	Atmel AVR	No	AT-MEGA32U4	8MHz	28KB	2.50KB
itsybitsy32u4_5V	Adafruit ItsyBitsy 5V/16MHz	Atmel AVR	No	AT-MEGA32U4	16MHz	28KB	2.50KB
metro	Adafruit Metro	Atmel AVR	No	AT-MEGA328P	16MHz	31.50KB	2KB
protrinket3	Adafruit Pro Trinket 3V/12MHz (USB)	Atmel AVR	No	AT-MEGA328P	12MHz	28KB	2KB
protrinket3ftdi	Adafruit Pro Trinket 3V/12MHz (FTDI)	Atmel AVR	No	AT-MEGA328P	12MHz	28KB	2KB
protrinket5	Adafruit Pro Trinket 5V/16MHz (USB)	Atmel AVR	No	AT-MEGA328P	16MHz	28KB	2KB
protrinket5ftdi	Adafruit Pro Trinket 5V/16MHz (FTDI)	Atmel AVR	No	AT-MEGA328P	16MHz	28KB	2KB
1.12. Embedded Boards	Adafruit Trinket 3V/8MHz	Atmel AVR	No	AT-TINY85	8MHz	8KB	512B
trinket5	Adafruit Trinket 5V/16MHz	Atmel AVR	No	AT-TINY85	16MHz	8KB	512B

1.12.6 Aiyarafun

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
node32s	Node32s	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

1.12.7 Alorium Technology

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
alorium_xlr8	Alorium XLR8	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB

1.12.8 Anarduino

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
miniwireless	Anarduino MiniWireless	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB

1.12.9 AppNearMe

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
micronfcboard	MicroNFCBoard	<i>NXP LPC</i>	No	LPC11U34	48MHz	48KB	10KB

1.12.10 April Brother

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espea32	April Brother ESPeA32	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

1.12.11 Arduboy

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
arduboy	Arduboy	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
arduboy_devkit	Arduboy DevKit	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB

1.12.12 Arduino

ID	Name	Platform	Debug	MCU
LilyPadUSB	Arduino LilyPad USB	<i>Atmel AVR</i>	No	ATMEGA32U4
atmega328pb	Atmel ATmega328PB	<i>Atmel AVR</i>	No	ATMEGA328PB
atmegangatmega168	Arduino NG or older ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
atmegangatmega8	Arduino NG or older ATmega8	<i>Atmel AVR</i>	No	ATMEGA8
btatmega168	Arduino BT ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
btatmega328	Arduino BT ATmega328	<i>Atmel AVR</i>	No	ATMEGA328P
chiwawa	Arduino Industrial 101	<i>Atmel AVR</i>	No	ATMEGA32U4
diecimilaatmega168	Arduino Duemilanove or Diecimila ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
diecimilaatmega328	Arduino Duemilanove or Diecimila ATmega328	<i>Atmel AVR</i>	No	ATMEGA328P
due	Arduino Due (Programming Port)	<i>Atmel SAM</i>	Yes	AT91SAM3X8E
dueUSB	Arduino Due (USB Native Port)	<i>Atmel SAM</i>	Yes	AT91SAM3X8E
esplora	Arduino Esplora	<i>Atmel AVR</i>	No	ATMEGA32U4
ethernet	Arduino Ethernet	<i>Atmel AVR</i>	No	ATMEGA328P
fio	Arduino Fio	<i>Atmel AVR</i>	No	ATMEGA328P
leonardo	Arduino Leonardo	<i>Atmel AVR</i>	No	ATMEGA32U4
leonardoeth	Arduino Leonardo ETH	<i>Atmel AVR</i>	No	ATMEGA32U4
lilypadatmega168	Arduino LilyPad ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
lilypadatmega328	Arduino LilyPad ATmega328	<i>Atmel AVR</i>	No	ATMEGA328P
megaADK	Arduino Mega ADK	<i>Atmel AVR</i>	No	ATMEGA2560
megaatmega1280	Arduino Mega or Mega 2560 ATmega1280	<i>Atmel AVR</i>	No	ATMEGA1280
megaatmega2560	Arduino Mega or Mega 2560 ATmega2560 (Mega 2560)	<i>Atmel AVR</i>	No	ATMEGA2560
micro	Arduino Micro	<i>Atmel AVR</i>	No	ATMEGA32U4
miniatmega168	Arduino Mini ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
miniatmega328	Arduino Mini ATmega328	<i>Atmel AVR</i>	No	ATMEGA328P
mkr1000USB	Arduino MKR1000	<i>Atmel SAM</i>	Yes	SAMD21G18A
mkrfox1200	Arduino MKR FOX 1200	<i>Atmel SAM</i>	Yes	SAMD21G18A
mkrgsm1400	Arduino MKR GSM 1400	<i>Atmel SAM</i>	Yes	SAMD21G18A
mkrwan1300	Arduino MKR WAN 1300	<i>Atmel SAM</i>	Yes	SAMD21G18A
mkrzero	Arduino MKRZERO	<i>Atmel SAM</i>	Yes	SAMD21G18A
mzeroUSB	Arduino M0	<i>Atmel SAM</i>	Yes	SAMD21G18A
mzeropro	Arduino M0 Pro (Programming/Debug Port)	<i>Atmel SAM</i>	Yes	SAMD21G18A
mzeroprouSB	Arduino M0 Pro (Native USB Port)	<i>Atmel SAM</i>	Yes	SAMD21G18A
nanoatmega168	Arduino Nano ATmega168	<i>Atmel AVR</i>	No	ATMEGA168
nanoatmega328	Arduino Nano ATmega328	<i>Atmel AVR</i>	No	ATMEGA328P
pro16MHzatmega168	Arduino Pro or Pro Mini ATmega168 (5V, 16 MHz)	<i>Atmel AVR</i>	No	ATMEGA168
pro16MHzatmega328	Arduino Pro or Pro Mini ATmega328 (5V, 16 MHz)	<i>Atmel AVR</i>	No	ATMEGA328P
pro8MHzatmega168	Arduino Pro or Pro Mini ATmega168 (3.3V, 8 MHz)	<i>Atmel AVR</i>	No	ATMEGA168
pro8MHzatmega328	Arduino Pro or Pro Mini ATmega328 (3.3V, 8 MHz)	<i>Atmel AVR</i>	No	ATMEGA328P
robotControl	Arduino Robot Control	<i>Atmel AVR</i>	No	ATMEGA32U4
robotMotor	Arduino Robot Motor	<i>Atmel AVR</i>	No	ATMEGA32U4
tian	Arduino Tian	<i>Atmel SAM</i>	Yes	SAMD21G18A
uno	Arduino Uno	<i>Atmel AVR</i>	No	ATMEGA328P
yun	Arduino Yun	<i>Atmel AVR</i>	No	ATMEGA32U4
yunmini	Arduino Yun Mini	<i>Atmel AVR</i>	No	ATMEGA32U4
zero	Arduino Zero (Programming/Debug Port)	<i>Atmel SAM</i>	Yes	SAMD21G18A
zeroUSB	Arduino Zero (USB Native Port)	<i>Atmel SAM</i>	Yes	SAMD21G18A

1.12.13 Armstrap

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
armstrap_eagle10	Armstrap Eagle 1024	ST STM32	Yes	STM32F417VGT	6168MHz	1MB	192KB
armstrap_eagle20	Armstrap Eagle 2048	ST STM32	Yes	STM32F427VIT	6168MHz	1.99MB	256KB
armstrap_eagle51	Armstrap Eagle 512	ST STM32	Yes	STM32F407VET	6168MHz	512KB	192KB

1.12.14 Atmel

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
attiny13	Generic ATTiny13	<i>Atmel AVR</i>	No	ATTINY13	9MHz	1KB	64B
attiny1634	Generic ATTiny1634	<i>Atmel AVR</i>	No	AT-TINY1634	8MHz	16KB	1KB
attiny167	Generic ATTiny167	<i>Atmel AVR</i>	No	ATTINY167	8MHz	16KB	512B
attiny2313	Generic ATTiny2313	<i>Atmel AVR</i>	No	AT-TINY2313	8MHz	2KB	128B
attiny24	Generic ATTiny24	<i>Atmel AVR</i>	No	ATTINY24	8MHz	2KB	128B
attiny25	Generic ATTiny25	<i>Atmel AVR</i>	No	ATTINY25	8MHz	2KB	128B
attiny261	Generic ATTiny261	<i>Atmel AVR</i>	No	ATTINY261	8MHz	2KB	128B
attiny4313	Generic ATTiny4313	<i>Atmel AVR</i>	No	AT-TINY4313	8MHz	4KB	256B
attiny44	Generic ATTiny44	<i>Atmel AVR</i>	No	ATTINY44	8MHz	4KB	256B
attiny441	Generic ATTiny441	<i>Atmel AVR</i>	No	ATTINY441	8MHz	4KB	256B
attiny45	Generic ATTiny45	<i>Atmel AVR</i>	No	ATTINY45	8MHz	4KB	256B
attiny461	Generic ATTiny461	<i>Atmel AVR</i>	No	ATTINY461	8MHz	4KB	256B
attiny48	Generic ATTiny48	<i>Atmel AVR</i>	No	ATTINY48	8MHz	4KB	256B
attiny828	Generic ATTiny828	<i>Atmel AVR</i>	No	ATTINY828	8MHz	8KB	512B
attiny84	Generic ATTiny84	<i>Atmel AVR</i>	No	ATTINY84	8MHz	8KB	512B
attiny841	Generic ATTiny841	<i>Atmel AVR</i>	No	ATTINY841	8MHz	8KB	512B
attiny85	Generic ATTiny85	<i>Atmel AVR</i>	No	ATTINY85	8MHz	8KB	512B
attiny861	Generic ATTiny861	<i>Atmel AVR</i>	No	ATTINY861	8MHz	8KB	512B
attiny87	Generic ATTiny87	<i>Atmel AVR</i>	No	ATTINY87	8MHz	8KB	512B
attiny88	Generic ATTiny88	<i>Atmel AVR</i>	No	ATTINY88	8MHz	8KB	512B
samd21_xpro	Atmel SAMD21-XPRO	<i>Atmel SAM</i>	Yes	SAMD21J18A	48MHz	256KB	32KB
samd21g18a	Atmel ATSAMW25-XPRO	<i>Atmel SAM</i>	Yes	SAMD21G18A	48MHz	256KB	32KB
saml21_xpro_b	Atmel SAML21-XPRO-B	<i>Atmel SAM</i>	Yes	SAML21J18B	48MHz	256KB	32KB
samr21_xpro	Atmel ATSAMR21-XPRO	<i>Atmel SAM</i>	Yes	SAMR21G18A	48MHz	256KB	32KB
usbasp	USBasp stick	<i>Atmel AVR</i>	No	ATMEGA8	12MHz	8KB	1KB

1.12.15 BBC

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
bbcmicrobit	BBC micro:bit	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	256KB	16KB
bbcmicrobit_b	BBC micro:bit B(\$130)	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	256KB	16KB

1.12.16 BQ

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
zumbt328	BQ ZUM BT-328	<i>Atmel AVR</i>	No	ATMEGA328P	16MHz	28KB	2KB

1.12.17 BitWizard

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
raspduino	BitWizard Raspduino	<i>Atmel AVR</i>	No	ATMEGA328P	16MHz	30KB	2KB

1.12.18 BluzDK

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
bluz_dk	BluzDK	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB

1.12.19 CQ Publishing

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
lpc11u35_501	CQ Publishing	TG-LPC11U35-501	<i>NXP LPC</i>	LPC11U35	48MHz	64KB	10KB

1.12.20 Controllino

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
controllino_maxi	Controllino Maxi	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	248KB	8KB
controllino_maxi_automation	Controllino Maxi Automation	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	248KB	8KB
controllino_mega	Controllino Mega	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	248KB	8KB
controllino_mini	Controllino Mini	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB

1.12.21 DFRobot

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
firebeetle32	FireBeetle-ESP32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

1.12.22 DOIT

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
esp32doit-devkit-v1	DOIT ESP32 DEVKIT V1	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

1.12.23 Delta

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
delta_dfbm_nq620	Delta DFBM-NQ620	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52832	64MHz	512KB	64KB
delta_dfcm_nnn50	Delta DFCM-NNN50	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	16KB
dfcm_nnn40	Delta DFCM-NNN40	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB

1.12.24 DigiStump

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
oak	DigiStump Oak	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

1.12.25 Digilent

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
cerebot32mx4	Digilent Cerebot 32MX4	<i>Microchip PIC32</i>	No	32MX460F512L	80MHz	508KB	32KB
cerebot32mx7	Digilent Cerebot 32MX7	<i>Microchip PIC32</i>	No	32MX795F512L	80MHz	508KB	128KB
chipkit_cmod	Digilent chipKIT Cmod	<i>Microchip PIC32</i>	No	32MX150F128D	40MHz	124KB	32KB
chipkit_dp32	Digilent chipKIT DP32	<i>Microchip PIC32</i>	No	32MX250F128B	40MHz	120KB	32KB
chipkit_mx3	Digilent chipKIT MX3	<i>Microchip PIC32</i>	No	32MX320F128H	80MHz	124KB	16KB
chipkit_pro_mx4	Digilent chipKIT Pro MX4	<i>Microchip PIC32</i>	No	32MX460F512L	80MHz	508KB	32KB
chipkit_pro_mx7	Digilent chipKIT Pro MX7	<i>Microchip PIC32</i>	No	32MX795F512L	80MHz	508KB	128KB
chipkit_uc32	Digilent chipKIT uC32	<i>Microchip PIC32</i>	No	32MX340F512H	80MHz	508KB	32KB
chipkit_wf32	Digilent chipKIT WF32	<i>Microchip PIC32</i>	No	32MX695F512L	80MHz	508KB	128KB
chipkit_wifi	Digilent chipKIT WiFire	<i>Microchip PIC32</i>	No	32MZ2048ECG1	100MHz	1.98MB	512KB
mega_pic32	Digilent chipKIT MAX32	<i>Microchip PIC32</i>	No	32MX795F512L	80MHz	508KB	128KB
openscope	Digilent OpenScope	<i>Microchip PIC32</i>	No	32MZ2048EFG1	2400MHz	1.98MB	512KB
uno_pic32	Digilent chipKIT UNO32	<i>Microchip PIC32</i>	No	32MX320F128H	80MHz	124KB	16KB

1.12.26 Digistump

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
digispark-pro	Digispark Pro	<i>Atmel AVR</i>	No	AT-TINY167	16MHz	14.50KB	512B
digispark-pro	Digispark Pro (32 byte buffer)	<i>Atmel AVR</i>	No	AT-TINY167	16MHz	14.50KB	512B
digispark-pro	Digispark Pro (16 MHz) (64 byte buffer)	<i>Atmel AVR</i>	No	AT-TINY167	16MHz	14.50KB	512B
digispark-tin	Digispark USB	<i>Atmel AVR</i>	No	ATTINY85	16MHz	5.87KB	512B
digix	Digistump DigiX	<i>Atmel SAM</i>	Yes	AT91SAM3X8B4MHz	512KB	28KB	

1.12.27 Doit

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
espduino	ESPDuino (ESP-13 Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

1.12.28 Dongsen Technology

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
pocket_32	Dongsen Tech Pocket 32	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

1.12.29 Dwengo

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
dwenguino	Dwenguino	<i>Atmel AVR</i>	No	AT90USB646	16MHz	60KB	2KB

1.12.30 DycodeX

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espectro	ESPectro Core	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
espectro32	ESPectro32	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

1.12.31 ESP32vn

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp32vn-iot-uno	ESP32vn IoT Uno	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

1.12.32 ESPert

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
espresso_lite_v1	ESPRESSO Lite 1.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
espresso_lite_v2	ESPRESSO Lite 2.0	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

1.12.33 ESPino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espino	ESPino	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

1.12.34 Electronic SweetPeas

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp320	Electronic SweetPeas ESP320	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

1.12.35 Elektor

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
elektor_uno_r4	Elektor Uno R4	<i>Atmel AVR</i>	No	AT-MEGA328PB	16MHz	31.50KB	2KB

1.12.36 Elektor Labs

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
elektor_cocorico	CoCo-ri-Co!	<i>NXP LPC</i>	Yes	LPC812	30MHz	16KB	4KB

1.12.37 Embedded Artists

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
lpc11u35	EA LPC11U35 QuickStart Board	<i>NXP LPC</i>	Yes	LPC11U35	48MHz	64KB	10KB
lpc4088	Embedded Artists LPC4088 Quick-Start Board	<i>NXP LPC</i>	Yes	LPC4088	120MHz	512KB	96KB
lpc4088_dm	Embedded Artists LPC4088 Display Module	<i>NXP LPC</i>	Yes	LPC4088	120MHz	512KB	96KB

1.12.38 Engduino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
engduinov3	Engduino 3	<i>Atmel AVR</i>	No	ATMEGA32U4	8MHz	28KB	2.50KB

1.12.39 EnviroDIY

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mayfly	EnviroDIY Mayfly	<i>Atmel AVR</i>	No	ATMEGA1284P	8MHz	127KB	16KB

1.12.40 Espotel

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
elmo_f411re	Espotel LoRa Mod- ule	<i>ST</i> <i>STM32</i>	<i>Yes</i>	STM32F411RET6	100MHz	512KB	128KB

1.12.41 Espressif

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
esp-wrover-kit	Espressif ESP-WROVER-KIT	<i>Espressif</i> 32	<i>Yes</i>	ESP32	240MHz	4MB	320KB
esp01	Espressif Generic ESP8266 ESP-01 512k	<i>Espressif</i> 8266	No	ESP8266	80MHz	512KB	80KB
esp01_1m	Espressif Generic ESP8266 ESP-01 1M	<i>Espressif</i> 8266	No	ESP8266	80MHz	1MB	80KB
esp07	Espressif Generic ESP8266 ESP-07	<i>Espressif</i> 8266	No	ESP8266	80MHz	4MB	80KB
esp12e	Espressif ESP8266 ESP-12E	<i>Espressif</i> 8266	No	ESP8266	80MHz	4MB	80KB
esp32dev	Espressif ESP32 Dev Module	<i>Espressif</i> 32	<i>Yes</i>	ESP32	240MHz	4MB	320KB
esp8285	Generic ESP8285 Module	<i>Espressif</i> 8266	No	ESP8266	80MHz	423.98KB	80KB
esp_wroom_02	ESP-WROOM-02	<i>Espressif</i> 8266	No	ESP8266	80MHz	2MB	80KB
phoenix_v1	Phoenix 1.0	<i>Espressif</i> 8266	No	ESP8266	80MHz	4MB	80KB
phoenix_v2	Phoenix 2.0	<i>Espressif</i> 8266	No	ESP8266	80MHz	4MB	80KB
pico32	ESP32 Pico Kit	<i>Espressif</i> 32	No	ESP32	240MHz	4MB	320KB
wifinfo	WifiInfo	<i>Espressif</i> 8266	No	ESP8266	80MHz	1MB	80KB

1.12.42 FPGAwars

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
icezum	IceZUM Alhambra	<i>Lattice</i> <i>iCE40</i>	No	ICE40-HX1K-TQ144	12MHz	32KB	32KB

1.12.43 Freescale

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
IBMEthernet	KEthernet IoT Starter Kit	<i>Freescale Kinetis</i>	Yes	MK64FN1M0VLL	120MHz	1MB	256KB
frdm_k20d50m	Freescale Kinetis FRDM-K20D50M	<i>Freescale Kinetis</i>	Yes	MK20DX128VLH	48MHz	128KB	16KB
frdm_k22f	Freescale Kinetis FRDM-K22F	<i>Freescale Kinetis</i>	Yes	MK22FN512VLH	120MHz	512KB	128KB
frdm_k64f	Freescale Kinetis FRDM-K64F	<i>Freescale Kinetis</i>	Yes	MK64FN1M0VLL	120MHz	1MB	256KB
frdm_k66f	Freescale Kinetis FRDM-K66F	<i>Freescale Kinetis</i>	Yes	MK66FN2M0VMD	80MHz	2MB	256KB
frdm_kl05z	Freescale Kinetis FRDM-KL05Z	<i>Freescale Kinetis</i>	Yes	MKL05Z32VFM	448MHz	32KB	4KB
frdm_kl25z	Freescale Kinetis FRDM-KL25Z	<i>Freescale Kinetis</i>	Yes	MKL25Z128VLK	448MHz	128KB	16KB
frdm_kl27z	Freescale Kinetis FRDM-KL27Z	<i>Freescale Kinetis</i>	Yes	MKL27Z64VLH	48MHz	64KB	16KB
frdm_kl43z	Freescale Kinetis FRDM-KL43Z	<i>Freescale Kinetis</i>	Yes	MKL43Z256VLH	448MHz	256KB	32KB
frdm_kl46z	Freescale Kinetis FRDM-KL46Z	<i>Freescale Kinetis</i>	Yes	MKL46Z256VLL	448MHz	256KB	32KB
frdm_kw41z	Freescale Kinetis FRDM-KW41Z	<i>Freescale Kinetis</i>	Yes	MKW41Z512VHT	48MHz	512KB	128KB

1.12.44 Fubarino

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
fubarino_mini	Fubarino Mini	<i>Microchip PIC32</i>	No	32MX250F128D	48MHz	120KB	32KB
fubarino_sd	Fubarino SD (1.5)	<i>Microchip PIC32</i>	No	32MX795F512H	80MHz	508KB	128KB

1.12.45 GHI Electronics

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
oc_mbuino	mBuino	<i>NXP LPC</i>	No	LPC11U24	50MHz	32KB	10KB

1.12.46 Generic

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
bluepill_f103c8	BluePill F103C8	ST STM32	Yes	STM32F103C8T6MHz	64KB	20KB	
genericSTM32F103CB	STM32F103C8 (20k RAM. 64k Flash)	ST STM32	Yes	STM32F103CBT6MHz	64KB	20KB	
genericSTM32F103CR	STM32F103CB (20k RAM. 128k Flash)	ST STM32	Yes	STM32F103CBT6MHz	128KB	20KB	
genericSTM32F103CR8	STM32F103R8 (20k RAM. 64 Flash)	ST STM32	Yes	STM32F103R8T6MHz	64KB	20KB	
genericSTM32F103RB	STM32F103RB (20k RAM. 128k Flash)	ST STM32	Yes	STM32F103RBT6MHz	128KB	20KB	
genericSTM32F103RC	STM32F103RC (48k RAM. 256k Flash)	ST STM32	Yes	STM32F103RCT6MHz	256KB	48KB	
genericSTM32F103RE	STM32F103RE (64k RAM. 512k Flash)	ST STM32	Yes	STM32F103RET6MHz	512KB	64KB	
genericSTM32F103VC	STM32F103VC (48k RAM. 256k Flash)	ST STM32	Yes	STM32F103VCT6MHz	256KB	48KB	
genericSTM32F103VE	STM32F103VE (64k RAM. 512k Flash)	ST STM32	Yes	STM32F103VET6MHz	512KB	64KB	
genericSTM32F407VE	STM32F407VE (192k RAM. 512k Flash)	ST STM32	Yes	STM32F407VEI68MHz	502.23KB	128KB	

1.12.47 Hardkernel

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
odroid_esp32	ODROID-GO	Espressif 32	No	ESP32	240MHz	16MB	320KB

1.12.48 Heltec

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
heltec_wifi_kit_8	Heltec Wifi kit 8	Espressif 8266	No	ESP8266	80MHz	4MB	80KB

1.12.49 Heltec Automation

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
heltec_wifi_kit_32	Heltec WIFI Kit 32	Espressif 32	No	ESP32	240MHz	4MB	320KB
heltec_wifi_lora_32	Heltec WIFI LoRa 32	Espressif 32	No	ESP32	240MHz	4MB	320KB

1.12.50 Hornbill

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
hornbill132dev	Hornbill ESP32 Dev	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB
hornbill132minima	Hornbill ESP32 Minima	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

1.12.51 Infineon

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
xmc1100_boot_kit	XMC1100 Boot Kit	<i>Infineon XMC</i>	Yes	XMC1100	32MHz	64KB	64KB
xmc1100_h_bridge2go	XMC1100 H-Bridge 2Go	<i>Infineon XMC</i>	Yes	XMC1100	32MHz	64KB	64KB
xmc1100_xmc2go	XMC1100 XMC2Go	<i>Infineon XMC</i>	Yes	XMC1100	32MHz	64KB	64KB
xmc1300_boot_kit	XMC1300 Boot Kit	<i>Infineon XMC</i>	Yes	XMC1300	32MHz	64KB	64KB
xmc1300_sense2go	XMC1300 Sense2GoL	<i>Infineon XMC</i>	Yes	XMC1300	32MHz	64KB	122.23KB
xmc4200_distance2go	XMC4200 Distance2Go	<i>Infineon XMC</i>	Yes	XMC4200	80MHz	250KB	256KB
xmc4700_relax_kit	XMC4700 Relax Kit	<i>Infineon XMC</i>	Yes	XMC4700	144MHz	2.00MB	1.95MB

1.12.52 Intel

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
genuino101	Arduino/Genuino 101	<i>Intel ARC32</i>	No	ARCV2EM	32MHz	152KB	80KB

1.12.53 IntoRobot

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
intorobot	IntoRobot Fig	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

1.12.54 JKSoft

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
wallbot_ble	JKSoft Wallbot BLE	Nordic nRF51	Yes	NRF51822	16MHz	128KB	16KB

1.12.55 Lattice

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
icestick	Lattice iCEstick FPGA Evaluation Kit	Lattice iCE40	No	ICE40-HX1K-TQ144	12MHz	32KB	32KB

1.12.56 LeafLabs

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
maple	Maple	ST STM32	Yes	STM32F103RBT6/2MHz	108KB	17KB	
maple_mini_b20	Maple Mini Boot-loader 2.0	ST STM32	Yes	STM32F103CBT6/2MHz	120KB	20KB	
maple_mini_orig	Maple Mini Original	ST STM32	Yes	STM32F103CBT6/2MHz	108KB	17KB	

1.12.57 LightUp

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lightup	LightUp	Atmel AVR	No	ATMEGA32U4	8MHz	28KB	2.50KB

1.12.58 Linino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
one	Linino One	Atmel AVR	No	ATMEGA32U4	16MHz	28KB	2.50KB

1.12.59 LowPowerLab

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mightyhat	LowPowerLab MightyHat	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31KB	2KB
moteino	LowPowerLab Moteino	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB
moteinomega	LowPowerLab MoteinoMEGA	<i>Atmel AVR</i>	No	AT-MEGA1284P	16MHz	127KB	16KB

1.12.60 M5Stack

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
m5stack-core-esp32	M5Stack Core ESP32	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB
m5stack-fire	M5Stack FIRE	<i>Espressif 32</i>	No	ESP32	240MHz	16MB	320KB

1.12.61 MH-ET Live

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mhetesp32devkit	MH ET ESP32DevKIT	LIVE <i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
mhetesp32minikit	MH ET ESP32MiniKit	LIVE <i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

1.12.62 MXChip

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mxchip_az31	Microsoft Azure IoT Development Kit (MXChip AZ3166)	<i>ST STM32</i>	<i>Yes</i>	STM32F412ZG	160MHz	1MB	256KB

1.12.63 Macchina

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
macchina2	Macchina M2	<i>Atmel SAM</i>	<i>Yes</i>	AT91SAM3X8E	84MHz	512KB	32KB

1.12.64 MakerAsia

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
nano32	MakerAsia Nano32	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

1.12.65 Maxim

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
max32600mbed	Maxim ARM mbed Enabled Development Platform for MAX32600	<i>Maxim 32</i>	Yes	MAX32600	4MHz	256KB	32KB
max32620hsp	Maxim Health Sensor Platform	<i>Maxim 32</i>	No	MAX32620	96MHz	2MB	256KB
max32625mbed	MAX32625MBED	<i>Maxim 32</i>	No	MAX32625	96MHz	512KB	160KB
max32625nexpaq	MAX32625NEXPAQ	<i>Maxim 32</i>	No	MAX32625	96MHz	512KB	160KB
max32630fthr	Maxim MAX32630FTHR Application Platform	<i>Maxim 32</i>	No	MAX32630	96MHz	2MB	512KB
maxwsnenv	Maxim Wireless Sensor Node Demonstrator	<i>Maxim 32</i>	Yes	MAX32610	4MHz	256KB	32KB

1.12.66 Mcudude

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mightycore1284	MightyCore mega1284	AT- <i>Atmel AVR</i>	No	AT-MEGA1284P	16MHz	127KB	16KB
mightycore16	MightyCore mega16	AT- <i>Atmel AVR</i>	No	ATMEGA16	16MHz	15.50KB	1KB
mightycore164	MightyCore mega164	AT- <i>Atmel AVR</i>	No	AT-MEGA164P	16MHz	15.50KB	1KB
mightycore32	MightyCore mega32	AT- <i>Atmel AVR</i>	No	ATMEGA32	16MHz	31.50KB	2KB
mightycore324	MightyCore mega324	AT- <i>Atmel AVR</i>	No	AT-MEGA324P	16MHz	31.50KB	2KB
mightycore644	MightyCore mega644	AT- <i>Atmel AVR</i>	No	AT-MEGA644P	16MHz	63KB	4KB
mightycore853	MightyCore mega8535	AT- <i>Atmel AVR</i>	No	ATMEGA16	16MHz	7.50KB	512B

1.12.67 MediaTek Labs

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
smart7688	LinkIt Smart 7688 Duo	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB

1.12.68 Microchip

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
at90pwm216	Atmel AT90PWM216	<i>Atmel AVR</i>	No	AT90PWM216	16MHz	16KB	1KB
at90pwm316	Atmel AT90PWM316	<i>Atmel AVR</i>	No	AT90PWM316	16MHz	16KB	1KB

1.12.69 Microduino

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
1284p16m	Microduino Core+ (AT-mega1284P@16M,5V)	<i>Atmel AVR</i>	No	AT-MEGA1284P	16MHz	127KB	16KB
1284p8m	Microduino Core+ (AT-mega1284P@8M,3.3V)	<i>Atmel AVR</i>	No	AT-MEGA1284P	8MHz	127KB	16KB
168pa16m	Microduino Core (At-mega168PA@16M,5V)	<i>Atmel AVR</i>	No	AT-MEGA168P	16MHz	15.50KB	1KB
168pa8m	Microduino Core (At-mega168PA@8M,3.3V)	<i>Atmel AVR</i>	No	AT-MEGA168P	8MHz	15.50KB	1KB
328p16m	Microduino Core (At-mega328P@16M,5V)	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB
328p8m	Microduino Core (At-mega328P@8M,3.3V)	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
32u416m	Microduino Core USB (AT-mega32U4@16M,5V)	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
644pa16m	Microduino Core+ (At-mega644PA@16M,5V)	<i>Atmel AVR</i>	No	AT-MEGA644P	16MHz	63KB	4KB
644pa8m	Microduino Core+ (At-mega644PA@8M,3.3V)	<i>Atmel AVR</i>	No	AT-MEGA644P	8MHz	63KB	4KB
microduino-core	Microduino Core ESP32	<i>Espres-sif 32</i>	No	ESP32	240MHz	4MB	320KB

1.12.70 Micromint

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lpc4330_m4	Bambino-210E	<i>NXP LPC</i>	<i>Yes</i>	LPC4330	204MHz	8MB	264KB

1.12.71 MikroElektronika

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
clicker2	MikroElektronika Clicker 2	<i>Microchip PIC32</i>	No	32MX460F512L	80MHz	508KB	32KB
flipnclikm	MikroElektronika Flip N Click MZ	<i>Microchip PIC32</i>	No	32MZ2048EFH100	252MHz	1.98MB	512KB
hexiwear	Hexiwear	<i>Freescale Kinetis</i>	Yes	MK64FN1M0VDC11	20MHz	1MB	256KB

1.12.72 MultiTech

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
mts_dragonfly_f411	MTS Dragonfly	<i>ST STM32</i>	Yes	STM32F411RET	6100MHz	512KB	128KB
mts_mdot_f405rg	MultiTech mDot	<i>ST STM32</i>	Yes	STM32F411RET	6100MHz	512KB	128KB
mts_mdot_f411re	MultiTech mDot F411	<i>ST STM32</i>	Yes	STM32F411RET	6100MHz	512KB	128KB
xdot_1151cc	MultiTech xDot	<i>ST STM32</i>	Yes	STM32L151CCU	62MHz	256KB	32KB

1.12.73 NGX Technologies

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
blueboard_lpc11u24	NGX Technologies BlueBoard-LPC11U24	<i>NXP LPC</i>	Yes	LPC11U24	448MHz	32KB	8KB

1.12.74 NXP

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
lpc11c24	NXP LPC11C24	<i>NXP LPC</i>	<i>Yes</i>	LPC11C24	48MHz	32KB	8KB
lpc11u24	NXP mbed LPC11U24	<i>NXP LPC</i>	<i>Yes</i>	LPC11U24	48MHz	32KB	8KB
lpc11u24_30	ARM mbed LPC11U24 (+CAN)	<i>NXP LPC</i>	<i>Yes</i>	LPC11U24	48MHz	32KB	8KB
lpc11u34_42	NXP LPC11U34	<i>NXP LPC</i>	<i>Yes</i>	LPC11U34	48MHz	40KB	8KB
lpc11u37_50	NXP LPC11U37	<i>NXP LPC</i>	<i>Yes</i>	LPC11U37	48MHz	128KB	10KB
lpc11u68	LPCXpresso11U68	<i>NXP LPC</i>	<i>Yes</i>	LPC11U68	50MHz	256KB	36KB
lpc1549	NXP LPCXpresso1549	<i>NXP LPC</i>	<i>Yes</i>	LPC1549	72MHz	256KB	36KB
lpc1768	NXP mbed LPC1768	<i>NXP LPC</i>	<i>Yes</i>	LPC1768	96MHz	512KB	64KB
lpc54114	NXP LPCXpresso54114	<i>NXP LPC</i>	<i>Yes</i>	LPC54114J256BD	6400MHz	256KB	192KB
lpc546xx	NXP LPCXpresso54608	<i>NXP LPC</i>	<i>Yes</i>	LPC54608ET512	180MHz	512KB	200KB
lpc812	NXP LPC800-MAX	<i>NXP LPC</i>	<i>Yes</i>	LPC812	30MHz	16KB	4KB
lpc824	LPCXpresso824-MAX	<i>NXP LPC</i>	<i>Yes</i>	LPC824	30MHz	32KB	8KB

1.12.75 NodeMCU

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
nodemcu	NodeMCU 0.9 (ESP-12 Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
nodemcu-32s	NodeMCU-32S	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
nodemcuv2	NodeMCU 1.0 (ESP-12E Module)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

1.12.76 Noduino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
quantum	Noduino Quantum	<i>Espressif 32</i>	No	ESP32	240MHz	16MB	320KB

1.12.77 Nordic

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
nrf51_dk	Nordic nRF51-DK	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB
nrf51_dongle	Nordic nRF51-Dongle	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB
nrf51_mkit	Nordic nRF51822-mKIT	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	128KB	16KB
nrf52840_dk	Nordic nRF52840-DK	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52840	64MHz	1MB	256KB
nrf52_dk	Nordic nRF52-DK	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

1.12.78 OLIMEX

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
esp32-evb	OLIMEX ESP32-EVB	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
esp32-gateway	OLIMEX ESP32-GATEWAY	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

1.12.79 OSHChip

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
oshchip	OSHChip	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB

1.12.80 Olimex

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
modwifi	Olimex MOD-WIFI-ESP8266(-DEV)	<i>Espressif 8266</i>	No	ESP8266	80MHz	2MB	80KB
pinguino32	Olimex PIC32-PINGUINO	<i>Microchip PIC32</i>	No	32MX440F256H	80MHz	252KB	32KB

1.12.81 Onehorse

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
onehorse32dev	Onehorse ESP32 Dev Module	<i>Espressif 32</i>	No	ESP32	240MHz	4MB	320KB

1.12.82 OpenBCI

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
openbci	OpenBCI 32bit	<i>Microchip PIC32</i>	No	32MX250F128B	40MHz	120KB	32KB

1.12.83 OpenEnergyMonitor

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
emonpi	OpenEnergyMonitor emonPi	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	30KB	2KB

1.12.84 Outrageous Circuits

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mbuino	Outrageous Circuits mBuino	<i>NXP LPC</i>	No	LPC11U24	48MHz	32KB	8KB

1.12.85 PONTECH

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
usbono_pic32	PONTECH UAV100	<i>Microchip PIC32</i>	No	32MX440F512H	80MHz	508KB	32KB

1.12.86 PanStamp

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
panStampAVR	PanStamp AVR	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
panStampNRG	PanStamp NRG 1.1	<i>TI MSP430</i>	No	CC430F5137	12MHz	31.88KB	4KB

1.12.87 Pinoccio

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
pinoccio	Pinoccio Scout	<i>Atmel AVR</i>	No	ATMEGA256RFR2	16MHz	248KB	32KB

1.12.88 Pololu Corporation

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
a-star32U4	Pololu A-Star 32U4	Atmel AVR	No	AT-MEGA32U4	16MHz	28KB	2.50KB

1.12.89 Pontech

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
nofire	Pontech NoFire	Microchip PIC32	No	32MZ2048EFG100 200MHz	1.98MB	512KB	
quick240_usb	Pontech Quick240	Microchip PIC32	No	32MX795F512L	80MHz	508KB	128KB

1.12.90 Punch Through

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
lightblue-bean	LightBlue Bean	Atmel AVR	No	AT-MEGA328P	8MHz	31.50KB	2KB
lightblue-beanplus	LightBlue Bean+	Atmel AVR	No	AT-MEGA328P	16MHz	31.50KB	2KB

1.12.91 Quirkbot

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
quirkbot	Quirkbot	Atmel AVR	No	ATMEGA32U4	8MHz	28KB	2.50KB

1.12.92 RFduino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
rfduino	RFduino	Nordic nRF51	Yes	NRF51822	16MHz	128KB	8KB

1.12.93 Raspberry Pi

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
raspberrypi_1b	Raspberry Pi 1 Model B	<i>Linux ARM</i>	No	BCM2835	700MHz	512MB	512MB
raspberrypi_2b	Raspberry Pi 2 Model B	<i>Linux ARM</i>	No	BCM2836	900MHz	1GB	1GB
raspberrypi_3b	Raspberry Pi 3 Model B	<i>Linux ARM</i>	No	BCM2837	1200MHz	1GB	1GB
raspberrypi_zero	Raspberry Pi Zero	<i>Linux ARM</i>	No	BCM2835	1000MHz	512MB	512MB

1.12.94 RedBearLab

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
blend	RedBearLab Blend	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
blendmicro16	RedBearLab Blend Micro 3.3V/16MHz (overclock)	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
blendmicro8	RedBearLab Blend Micro 3.3V/8MHz	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
redBearLab	RedBearLab nRF51822	<i>Nordic nRF51</i>	Yes	NRF51822	16MHz	256KB	16KB
redBearLabBLE	RedBearLab BLE Nano 1.5	<i>Nordic nRF51</i>	Yes	NRF51822	16MHz	256KB	32KB
redbear_blend	RedBearLab BLE Nano 2	<i>Nordic nRF52</i>	Yes	NRF52832	64MHz	512KB	64KB
redbear_blend	RedBearLab Blend 2	<i>Nordic nRF52</i>	Yes	NRF52832	64MHz	512KB	64KB

1.12.95 RepRap

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
reprap_rambo	RepRap RAMBo	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	252KB	8KB

1.12.96 RoboticsBrno

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
alksesp32	ALKS ESP32	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

1.12.97 RushUp

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
cloud_jam	RushUp Cloud-JAM	ST STM32	Yes	STM32F401RET6	84MHz	512KB	96KB
cloud_jam_14	RushUp Cloud-JAM L4	ST STM32	Yes	STM32L476RGT6	80MHz	1MB	128KB
kitra_520	RushUp Kitra 520	Linux ARM	No	EXYNOS3250	1000MHz	4GB	512MB

1.12.98 SODAQ

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
sodaq_autonomo	SODAQ Autonomo	Atmel SAM	Yes	SAMD21J18A	48MHz	256KB	32KB
sodaq_explorer	SODAQ Ex-pLoRer	Atmel SAM	Yes	SAMD21J18A	48MHz	256KB	32KB
sodaq_galora	SODAQ GaLoRa	Atmel AVR	No	AT-MEGA1284P	8MHz	127KB	16KB
sodaq_mbili	SODAQ Mbili	Atmel AVR	No	AT-MEGA1284P	8MHz	127KB	16KB
sodaq_moja	SODAQ Moja	Atmel AVR	No	AT-MEGA328P	8MHz	31.50KB	2KB
sodaq_ndogo	SODAQ Ndogo	Atmel AVR	No	AT-MEGA1284P	8MHz	127KB	16KB
sodaq_one	SODAQ ONE	Atmel SAM	Yes	SAMD21G18A	48MHz	256KB	32KB
sodaq_tatu	SODAQ Tatu	Atmel AVR	No	AT-MEGA1284P	8MHz	127KB	16KB

1.12.99 ST

ID	Name	Platform	Debug	MCU	Frequency	Flash
disco_f030r8	ST STM32F0308DISCOVERY	ST STM32	Yes	STM32F030R8T6	48MHz	64KB
disco_f051r8	ST STM32F0DISCOVERY	ST STM32	Yes	STM32F051R8T6	48MHz	64KB
disco_f100rb	ST STM32VLDISCOVERY	ST STM32	Yes	STM32F100RBT6	24MHz	128KB
disco_f303vc	ST STM32F3DISCOVERY	ST STM32	Yes	STM32F303VCT6	72MHz	256KB
disco_f334c8	ST 32F3348DISCOVERY	ST STM32	Yes	STM32F334C8T6	72MHz	64KB
disco_f401vc	ST 32F401CDISCOVERY	ST STM32	Yes	STM32F401VCT6	84MHz	256KB
disco_f407vg	ST STM32F4DISCOVERY	ST STM32	Yes	STM32F407VGT6	168MHz	1MB
disco_f411ve	ST 32F411EDISCOVERY	ST STM32	Yes	STM32F411VET6	100MHz	512KB
disco_f413zh	ST 32F413HDISCOVERY	ST STM32	Yes	STM32F413ZHT6	100MHz	512KB
disco_f429zi	ST 32F429IDISCOVERY	ST STM32	Yes	STM32F429ZIT6	180MHz	2MB
disco_f469ni	ST 32F469IDISCOVERY	ST STM32	Yes	STM32F469NIH6	180MHz	1MB
disco_f746ng	ST 32F746GDISCOVERY	ST STM32	Yes	STM32F746NGH6	216MHz	1MB
disco_f769ni	ST 32F769IDISCOVERY	ST STM32	Yes	STM32F769NIH6	80MHz	1MB

Continued on

Table 12 – continued from previous page

ID	Name	Platform	Debug	MCU	Frequency	Flash
disco_1053c8	ST 32L0538DISCOVERY	ST STM32	Yes	STM32L053C8T6	32MHz	64KB
disco_1072cz_lrwan1	ST DISCO-L072CZ-LRWAN1	ST STM32	Yes	STM32L072CZ	32MHz	192KB
disco_1152rb	ST STM32LDISCOVERY	ST STM32	Yes	STM32L152RBT6	32MHz	128KB
disco_1475vg_iot01a	ST DISCO-L475VG-IOT01A	ST STM32	Yes	STM32L475VGT6	80MHz	1MB
disco_1476vg	ST 32L476GDISCOVERY	ST STM32	Yes	STM32L476VGT6	80MHz	1MB
eval_1073z	ST STM32L073Z-EVAL	ST STM32	Yes	STM32L073VZT6	32MHz	192KB
nucleo_f030r8	ST Nucleo F030R8	ST STM32	Yes	STM32F030R8T6	48MHz	64KB
nucleo_f031k6	ST Nucleo F031K6	ST STM32	Yes	STM32F031K6T6	48MHz	32KB
nucleo_f042k6	ST Nucleo F042K6	ST STM32	Yes	STM32F042K6T6	48MHz	32KB
nucleo_f070rb	ST Nucleo F070RB	ST STM32	Yes	STM32F070RBT6	48MHz	128KB
nucleo_f072rb	ST Nucleo F072RB	ST STM32	Yes	STM32F072RBT6	48MHz	128KB
nucleo_f091rc	ST Nucleo F091RC	ST STM32	Yes	STM32F091RCT6	48MHz	256KB
nucleo_f103rb	ST Nucleo F103RB	ST STM32	Yes	STM32F103RBT6	72MHz	128KB
nucleo_f207zg	ST Nucleo F207ZG	ST STM32	Yes	STM32F207ZGT6	120MHz	1MB
nucleo_f302r8	ST Nucleo F302R8	ST STM32	Yes	STM32F302R8T6	72MHz	64KB
nucleo_f303k8	ST Nucleo F303K8	ST STM32	Yes	STM32F303K8T6	72MHz	64KB
nucleo_f303re	ST Nucleo F303RE	ST STM32	Yes	STM32F303RET6	72MHz	512KB
nucleo_f303ze	ST Nucleo F303ZE	ST STM32	Yes	STM32F303ZET6	72MHz	512KB
nucleo_f334r8	ST Nucleo F334R8	ST STM32	Yes	STM32F334R8T6	72MHz	64KB
nucleo_f401re	ST Nucleo F401RE	ST STM32	Yes	STM32F401RET6	84MHz	512KB
nucleo_f410rb	ST Nucleo F410RB	ST STM32	Yes	STM32F410RBT6	100MHz	128KB
nucleo_f411re	ST Nucleo F411RE	ST STM32	Yes	STM32F411RET6	100MHz	512KB
nucleo_f412zg	ST Nucleo F412ZG	ST STM32	Yes	STM32F412ZGT6	100MHz	1MB
nucleo_f413zh	ST Nucleo F413ZH	ST STM32	Yes	STM32F413ZHT6	100MHz	512KB
nucleo_f429zi	ST Nucleo F429ZI	ST STM32	Yes	STM32F429ZIT6	180MHz	2MB
nucleo_f446re	ST Nucleo F446RE	ST STM32	Yes	STM32F446RET6	180MHz	512KB
nucleo_f446ze	ST Nucleo F446ZE	ST STM32	Yes	STM32F446ZET6	180MHz	512KB
nucleo_f746zg	ST Nucleo F746ZG	ST STM32	Yes	STM32F746ZGT6	216MHz	1MB
nucleo_f767zi	ST Nucleo F767ZI	ST STM32	Yes	STM32F767ZIT6	216MHz	2MB
nucleo_1011k4	ST Nucleo L011K4	ST STM32	Yes	STM32L011K4T6	32MHz	16KB
nucleo_1031k6	ST Nucleo L031K6	ST STM32	Yes	STM32L031K6T6	32MHz	32KB
nucleo_1053r8	ST Nucleo L053R8	ST STM32	Yes	STM32L053R8T6	32MHz	64KB
nucleo_1073rz	ST Nucleo L073RZ	ST STM32	Yes	STM32L073RZ	32MHz	192KB
nucleo_1152re	ST Nucleo L152RE	ST STM32	Yes	STM32L152RET6	32MHz	512KB
nucleo_1432kc	ST Nucleo L432KC	ST STM32	Yes	STM32L432KCU6	80MHz	256KB
nucleo_1476rg	ST Nucleo L476RG	ST STM32	Yes	STM32L476RGT6	80MHz	1MB
nucleo_1496zg	ST Nucleo L496ZG	ST STM32	Yes	STM32L496ZGT6	80MHz	1MB

1.12.100 SainSmart

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
sainSmartDue	SainSmart Due (Programming Port)	Atmel SAM	Yes	AT91SAM3X8E	4MHz	512KB	32KB
sainSmartDueUSB	SainSmart Due (USB Native Port)	Atmel SAM	Yes	AT91SAM3X8E	4MHz	512KB	32KB

1.12.101 Samsung

ID	Name	Platform	De- bug	MCU	Fre- quency	Flash	RAM
artik_053	Samsung ARTIK053	<i>Samsung ARTIK</i>	Yes	S5JT200	320MHz	8MB	1.25MB
artik_1020	Samsung ARTIK1020	<i>Linux ARM</i>	No	EXYNOS5422	1500MHz	16GB	2GB
artik_520	Samsung ARTIK520	<i>Linux ARM</i>	No	EXYNOS3250	1000MHz	4GB	512MB
artik_530	Samsung ARTIK530	<i>Linux ARM</i>	No	S5P4418	1200MHz	4GB	512MB
artik_710	Samsung ARTIK710	<i>Linux ARM</i>	No	S5P6818	1400MHz	4GB	1GB

1.12.102 Sanguino

ID	Name	Plat- form	De- bug	MCU	Fre- quency	Flash	RAM
sanguino_atmega1284p	Sanguino ATmega1284p (8MHz)	<i>Atmel AVR</i>	No	AT-MEGA1284P	8MHz	127KB	16KB
sanguino_atmega1284p	Sanguino ATmega1284p (16MHz)	<i>Atmel AVR</i>	No	AT-MEGA1284P	16MHz	127KB	16KB
sanguino_atmega644	Sanguino ATmega644 or ATmega644A (16 MHz)	<i>Atmel AVR</i>	No	AT-MEGA644	16MHz	63KB	4KB
sanguino_atmega644	Sanguino ATmega644 or ATmega644A (8 MHz)	<i>Atmel AVR</i>	No	AT-MEGA644	8MHz	63KB	4KB
sanguino_atmega644p	Sanguino ATmega644P or ATmega644PA (16 MHz)	<i>Atmel AVR</i>	No	AT-MEGA644P	16MHz	63KB	4KB
sanguino_atmega644p	Sanguino ATmega644P or ATmega644PA (8 MHz)	<i>Atmel AVR</i>	No	AT-MEGA644P	8MHz	63KB	4KB

1.12.103 SeeedStudio

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
cui32stem	SeeedStudio CUI32stem	<i>Microchip PIC32</i>	No	32MX795F512H	80MHz	508KB	128KB
seeedArchBLE	Seeed Arch BLE	<i>Nordic nRF51</i>	Yes	NRF51822	16MHz	128KB	16KB
seeedArchGPRS	Seeed Arch GPRS V2	<i>NXP LPC</i>	No	LPC11U37	48MHz	128KB	10KB
seeedArchLink	Seeed Arch Link	<i>Nordic nRF51</i>	Yes	NRF51822	16MHz	256KB	16KB
seeedArchMax	Seeed Arch Max	<i>ST STM32</i>	Yes	STM32F407VET6	168MHz	512KB	192KB
seeedArchPro	Seeed Arch Pro	<i>NXP LPC</i>	Yes	LPC1768	96MHz	512KB	64KB
seeedTinyBLE	Seeed Tiny BLE	<i>Nordic nRF51</i>	Yes	NRF51822	16MHz	256KB	16KB
seeeduino	Seeeduino	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB
wio_node	Wio Node	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
xadow_m0	Seeed Xadow M0	<i>NXP LPC</i>	No	LPC11U35	48MHz	64KB	10KB

1.12.104 Semtech

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mote_1152rc	NAMote72	<i>ST STM32</i>	Yes	STM32L152RC	32MHz	256KB	32KB

1.12.105 SiFive

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
freedom-e300-hifive1	HiFive1	<i>RISC-V</i>	Yes	FE310	320MHz	16MB	16KB

1.12.106 Silicon Labs

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
efm32gg_st	Silicon Labs EFM32GG-STK3700 (Giant Gecko)	Silicon Labs EFM32	Yes	EFM32GG990	F _{40MHz}	1MB	128KB
efm32hg_st	Silicon Labs SLSTK3400A USB-enabled (Happy Gecko)	Silicon Labs EFM32	Yes	EFM32HG322	F _{4MHz}	64KB	8KB
efm32lg_st	Silicon Labs EFM32LG-STK3600 (Leopard Gecko)	Silicon Labs EFM32	Yes	EFM32LG990	F _{25MHz}	256KB	32KB
efm32pg_st	Silicon Labs SLSTK3401A (Pearl Gecko)	Silicon Labs EFM32	Yes	EFM32PG1B200	F _{15MHz}	256KB	32KB
efm32wg_st	Silicon Labs EFM32WG-STK3800 (Wonder Gecko)	Silicon Labs EFM32	Yes	EFM32WG990	F _{25MHz}	256KB	32KB
efm32zg_st	Silicon Labs EFM32ZG-STK3200 (Zero Gecko)	Silicon Labs EFM32	Yes	EFM32ZG222	F _{24MHz}	32KB	4KB

1.12.107 Smeshlink

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
xbed_lpc1768	Smeshlink xbed	NXP LPC	No	LPC1768	96MHz	512KB	32KB

1.12.108 Solder Splash Labs

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
dipcortexm0	Solder Splash Labs DipCortex M0	NXP LPC	Yes	LPC11U24	50MHz	32KB	8KB
lpc1347	DipCortex M3	NXP LPC	Yes	LPC1347	72MHz	64KB	12KB

1.12.109 SparkFun

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
sparkfunBlynk	SparkFun Blynk Board	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
sparkfun_digital sandbox	SparkFun Digital Sandbox	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
sparkfun_fiov3	SparkFun Fio V3 3.3V/8MHz	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
sparkfun_makeymakey	SparkFun Makey Makey	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
sparkfun_megaminis3.3V	SparkFun Mega Pro Mini 3.3V	<i>Atmel AVR</i>	No	AT-MEGA2560	8MHz	252KB	8KB
sparkfun_megapro5V/16MHz	SparkFun Mega Pro 5V/16MHz	<i>Atmel AVR</i>	No	AT-MEGA2560	16MHz	248KB	8KB
sparkfun_megapro3.3V/8MHz	SparkFun Mega Pro 3.3V/8MHz	<i>Atmel AVR</i>	No	AT-MEGA2560	8MHz	252KB	8KB
sparkfun_promicro5V/16MHz	SparkFun Pro Micro 5V/16MHz	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB
sparkfun_promicro3.3V/8MHz	SparkFun Pro Micro 3.3V/8MHz	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
sparkfun_qduinomini	SparkFun Qduino Mini	<i>Atmel AVR</i>	No	AT-MEGA32U4	8MHz	28KB	2.50KB
sparkfun_redboard	SparkFun RedBoard	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB
sparkfun_samd21_development	SparkFun SAMD21 Dev Breakout	<i>Atmel SAM</i>	Yes	SAMD21G18A48MHz	256KB	32KB	
sparkfun_samd21_mini	SparkFun SAMD21 Mini Breakout	<i>Atmel SAM</i>	Yes	SAMD21G18A48MHz	256KB	32KB	
sparkfun_satmega128rfa1	SparkFun AT-mega128RFA1 Dev Board	<i>Atmel AVR</i>	No	AT-MEGA128RFA1	16MHz	16KB	124KB
sparkfun_serial7segment	SparkFun Serial 7-Segment Display	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	31.50KB	2KB
thing	SparkFun ESP8266 Thing	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB
thingdev	SparkFun ESP8266 Thing Dev	<i>Espressif 8266</i>	No	ESP8266	80MHz	512KB	80KB
uvview	SparkFun MicroView	<i>Atmel AVR</i>	No	AT-MEGA328P	16MHz	31.50KB	2KB

1.12.110 SparkFun Electronics

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp32thing	SparkFun ESP32 Thing	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

1.12.111 SpellFoundry

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
sleepypypi	SpellFoundry Sleepy Pi 2	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	30KB	2KB

1.12.112 SweetPea

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp210	SweetPea ESP-210	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

1.12.113 Switch Science

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
hrm1017	Switch Science mbed HRM1017	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	16MHz	256KB	16KB
lpc1114fn28	Switch Science mbed LPC1114FN28	<i>NXP LPC</i>	<i>Yes</i>	LPC1114FN28	48MHz	32KB	4KB
ssci824	Switch Science mbed LPC824	<i>NXP LPC</i>	<i>Yes</i>	LPC824	30MHz	32KB	8KB
ty51822r3	Switch Science mbed TY51822r3	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB

1.12.114 TI

ID	Name	Plat-form	De-bug	MCU	Fre-quency	Flash	RAM
1plm4f120h5q	TI LaunchPad (Stellaris) w/ lm4f120 (80MHz)	<i>TI TIVA</i>	<i>Yes</i>	LPLM4F120H5Q	80MHz	256KB	32KB
1pmmsp430f552	TI LaunchPad MSP-EXP430F5529LP	<i>TI MSP430</i>	<i>Yes</i>	MSP430F5529	16MHz	128KB	8KB
1pmmsp430fr41	TI LaunchPad MSP-EXP430FR4133LP	<i>TI MSP430</i>	<i>Yes</i>	MSP430FR4133	8MHz	15KB	2KB
1pmmsp430fr57	TI FraunchPad MSP-EXP430FR5739LP	<i>TI MSP430</i>	<i>Yes</i>	MSP430FR5739	16MHz	16KB	512B
1pmmsp430fr59	TI LaunchPad MSP-EXP430FR5969LP	<i>TI MSP430</i>	<i>Yes</i>	MSP430FR5969	8MHz	64KB	2KB
1pmmsp430fr69	TI LaunchPad MSP-EXP430FR6989LP	<i>TI MSP430</i>	<i>Yes</i>	MSP430FR6989	8MHz	127KB	2KB
1pmmsp430g255	TI LaunchPad MSP-EXP430G2553LP	<i>TI MSP430</i>	<i>Yes</i>	MSP430G2553	16MHz	16KB	512B
1ptm4c1230c3	Ti LaunchPad (Tiva C) w/ tm4c123 (80MHz)	<i>TI TIVA</i>	<i>Yes</i>	LPTM4C1230C3	80MHz	256KB	32KB
1ptm4c1294ncp	Ti LaunchPad (Tiva C) w/ tm4c129 (120MHz)	<i>TI TIVA</i>	<i>Yes</i>	LPTM4C1294NC	120MHz	1MB	256KB

1.12.115 TTGO

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
ttgo-lora32-v1	TTGO LoRa32-OLED V1	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

1.12.116 Taida Century

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
stct_nrf52_mini	Taida Century nRF52 mini board	<i>Nordic nRF52</i>	<i>Yes</i>	NRF52832	64MHz	512KB	64KB

1.12.117 Teensy

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
teensy20	Teensy 2.0	<i>Teensy</i>	No	ATMEGA32U4	16MHz	31.50KB	2.50KB
teensy20pp	Teensy++ 2.0	<i>Teensy</i>	No	AT90USB1286	16MHz	127KB	8KB
teensy30	Teensy 3.0	<i>Teensy</i>	No	MK20DX128	48MHz	128KB	16KB
teensy31	Teensy 3.1 / 3.2	<i>Teensy</i>	<i>Yes</i>	MK20DX256	72MHz	256KB	64KB
teensy35	Teensy 3.5	<i>Teensy</i>	<i>Yes</i>	MK64FX512	120MHz	512KB	192KB
teensy36	Teensy 3.6	<i>Teensy</i>	<i>Yes</i>	MK66FX1M0	180MHz	1MB	256KB
teensylc	Teensy LC	<i>Teensy</i>	<i>Yes</i>	MKL26Z64	48MHz	62KB	8KB

1.12.118 ThaiEasyElec

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espino32	ESPino32	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB
espinotee	ThaiEasyElec ESPino	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB

1.12.119 The Things Network

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
the_things_uno	The Things Uno	<i>Atmel AVR</i>	No	AT-MEGA32U4	16MHz	28KB	2.50KB

1.12.120 TinyCircuits

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
tinyduino	TinyCircuits TinyDuino Processor Board	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	30KB	2KB
tinylily	TinyCircuits TinyLily Mini Processor	<i>Atmel AVR</i>	No	AT-MEGA328P	8MHz	30KB	2KB

1.12.121 UBW32

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
ubw32_mx460	UBW32 MX460	<i>Microchip PIC32</i>	No	32MX460F512L	80MHz	508KB	32KB
ubw32_mx795	UBW32 MX795	<i>Microchip PIC32</i>	No	32MX795F512L	80MHz	508KB	128KB

1.12.122 VNG

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
vbluno51	VNG VBLUNO51	<i>Nordic nRF51</i>	Yes	NRF51822	16MHz	128KB	32KB

1.12.123 WEMOS

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
d1	WEMOS D1 R1 (Retired)	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
d1_mini	WeMos D1 R2 & mini	<i>Espressif 8266</i>	No	ESP8266	80MHz	4MB	80KB
d1_mini_lite	WeMos D1 mini Lite	<i>Espressif 8266</i>	No	ESP8266	80MHz	1MB	80KB
d1_mini_pro	WeMos D1 mini Pro	<i>Espressif 8266</i>	No	ESP8266	80MHz	16MB	80KB
lolin32	WEMOS LOLIN32	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB
lolin_d32	WEMOS LOLIN D32	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB
lolin_d32_pro	WEMOS LOLIN D32 PRO	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB
wemosbat	WeMos WiFi & Bluetooth Battery	<i>Espressif 32</i>	Yes	ESP32	240MHz	4MB	320KB

1.12.124 WIZNet

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
wizwiki_w7500	WIZwiki-W7500	<i>WIZNet W7500</i>	<i>Yes</i>	WIZNET7500	48MHz	128KB	48KB
wizwiki_w7500eco	WIZwiki-W7500ECO	<i>WIZNet W7500</i>	<i>Yes</i>	WIZ-NET7500ECO	48MHz	128KB	48KB
wizwiki_w7500p	WIZwiki-W7500P	<i>WIZNet W7500</i>	<i>Yes</i>	WIZ-NET7500P	48MHz	128KB	48KB

1.12.125 Waveshare

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
waveshare_ble400	Waveshare BLE400	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB

1.12.126 Wicked Device

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
wildfirev2	Wicked Device WildFire V2	<i>Atmel AVR</i>	No	AT-MEGA1284P	16MHz	120.00KB	16KB
wildfirev3	Wicked Device WildFire V3	<i>Atmel AVR</i>	No	AT-MEGA1284P	16MHz	127KB	16KB

1.12.127 Widora

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
widora-air	Widora AIR	<i>Espressif 32</i>	No	ESP32	240MHz	16MB	320KB

1.12.128 Xilinx

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
coreplexip-e31-artix7	Freedom E310 Arty (Artix-7) FPGA Dev Kit	<i>RISC-V</i>	<i>Yes</i>	E31	320MHz	16MB	256MB
coreplexip-e51-artix7	Freedom E51 Arty (Artix-7) FPGA Dev Kit	<i>RISC-V</i>	<i>Yes</i>	E51	1500MHz	16MB	256MB

1.12.129 XinaBox

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
xinabox_cw02	XinaBox CW02	<i>Espressif 32</i>	<i>Yes</i>	ESP32	240MHz	4MB	320KB

1.12.130 chipKIT

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lenny	chipKIT Lenny	<i>Microchip PIC32</i>	No	32MX270F256D	40MHz	120KB	32KB

1.12.131 element14

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
chipkit_pi	Element14 chipKIT Pi	<i>Microchip PIC32</i>	No	32MX250F128B	40MHz	120KB	32KB

1.12.132 makerlab.mx

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
altair	Altair	<i>Atmel AVR</i>	No	ATMEGA256RFR2	16MHz	248KB	32KB

1.12.133 ng-beacon

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
ng_beacon	ng-beacon	<i>Nordic nRF51</i>	<i>Yes</i>	NRF51822	32MHz	256KB	32KB

1.12.134 nicai-systems

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
bob3	nicai-systems BOB3 coding bot	<i>Atmel AVR</i>	No	AT-MEGA88	8MHz	8KB	1KB
nibo2	nicai-systems NIBO 2 robot	<i>Atmel AVR</i>	No	AT-MEGA128	16MHz	128KB	4KB
nibobee	nicai-systems NIBObee robot	<i>Atmel AVR</i>	No	AT-MEGA16	15MHz	16KB	1KB
nibobee_1284	nicai-systems NIBObee robot with Tuning Kit	<i>Atmel AVR</i>	No	AT-MEGA1284P	20MHz	128KB	16KB
niboburger	nicai-systems NIBO burger robot	<i>Atmel AVR</i>	No	AT-MEGA16	15MHz	16KB	1KB
niboburger_1284	nicai-systems NIBO burger robot with Tuning Kit	<i>Atmel AVR</i>	No	AT-MEGA1284P	20MHz	128KB	16KB

1.12.135 u-blox

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
mbed_connect_odin	u-blox Connect Cloud	ST STM32	Yes	STM32F439ZI	168MHz	2MB	256KB
mtb_ublox_odin	u-blox ODIN-W2	ST STM32	Yes	STM32F439ZI	168MHz	2MB	256KB
nina_w10	u-blox NINA-W10 series	Espressif 32	No	ESP32	240MHz	2MB	320KB
ublox_c030_n21	u-blox C030-N211 IoT Starter Kit	ST STM32	Yes	STM32F437VG	180MHz	1MB	256KB
ublox_c030_u20	u-blox C030-U201 IoT Starter Kit	ST STM32	Yes	STM32F437VG	180MHz	1MB	256KB
ublox_evk_nina	u-blox EVK-NINA-B1	Nordic nRF52	Yes	NRF52832	64MHz	512KB	64KB
ublox_evk_odin	u-blox EVK-ODIN-W2	ST STM32	Yes	STM32F439ZI	168MHz	2MB	256KB
ubloxc027	u-blox C027	NXP LPC	Yes	LPC1768	96MHz	512KB	64KB

1.12.136 ubIQio

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
ardhat	ubIQio Ardhata	Atmel AVR	No	ATMEGA328P	16MHz	31.50KB	2KB

1.12.137 y5 design

ID	Name	Platform	De-bug	MCU	Fre-quency	Flash	RAM
lpc11u35_y5_mbug	y5 LPC11U35 mbug	NXP LPC	Yes	LPC11U35	48MHz	64KB	10KB
nrf51822_y5_mbug	y5 nRF51822 mbug	Nordic nRF51	Yes	NRF51822	16MHz	256KB	16KB

1.13 Custom Platform & Board

1.13.1 Custom Development Platform

PlatformIO was developed like a tool that may build the same source code for the different development platforms via single command `platformio run` without any dependent software or requirements.

For this purpose *PlatformIO* uses own pre-configured platforms data: build scripts, toolchains, the settings for the most popular embedded boards and etc. These data are pre-built and packaged to the different packages. It allows *PlatformIO* to have multiple development platforms which can use the same packages(toolchains, frameworks), but have different/own build scripts, uploader and etc.

Note: If you want to change some build flags for the existing *Development Platforms*, you don't need to create (or duplicate) own development platforms! Please use `build_flags` option.

Step-by-Step Manual

1. Choose *Packages* for platform
2. Create *Manifest File* `platform.json`
3. Create *Build Script* `main.py`
4. Finish with the *Installation*.

Contents

- *Custom Development Platform*
 - *Packages*
 - *Manifest File* `platform.json`
 - *Build Script* `main.py`
 - *Installation*
 - *Examples*

Packages

PlatformIO has own registry with pre-built packages for the most popular systems and you can use them in your manifest. These packages are stored in super-fast and reliable CDN storage provided by JFrog Bintray:

- <https://bintray.com/platformio/dl-packages>

Manifest File `platform.json`

See example with a manifest for packages:

- <http://dl.platformio.org/packages/manifest.json>

```
{  
  "name": "myplatform",  
  "title": "My Platform",  
  "description": "My custom development platform",  
  "url": "http://example.com",  
  "homepage": "https://platformio.org/platforms/myplatform",  
  "license": "Apache-2.0",  
  "engines": {  
    "platformio": "~3.0.0"  
  },  
  "repository": {  
    "type": "git",  
    "url": "https://github.com/platformio/platform-myplatform.git"  
  },  
  "version": "0.0.0",  
  "packageRepositories": [  
    "https://dl.bintray.com/platformio/dl-packages/manifest.json",  
    "http://dl.platformio.org/packages/manifest.json",  
    {  
      "my_custom_package": [  
        {  
          "name": "myplatform",  
          "version": "0.0.0",  
          "url": "https://github.com/platformio/platform-myplatform.git"  
        }  
      ]  
    }  
  ]  
}
```

(continues on next page)

(continued from previous page)

```

"url": "http://dl.example.com/my_custom_package-darwin_x86_64-1.2.3.tar.gz",
"sha1": "bb7ddac56a314b5cb1926cc1790ae4de3a03e65c",
"version": "1.2.3",
"system": [
    "darwin_x86_64",
    "darwin_i386"
],
},
{
"url": "http://dl.example.com/my_custom_package-linux_aarch64-1.2.3.tar.gz",
"sha1": "127ddac56a314b5cb1926cc1790ae4de3a03e65c",
"version": "1.2.3",
"system": "linux_aarch64"
},
],
"framework-%FRAMEWORK_NAME_1%": [
{
"url":_
→"http://dl.example.com/packages/framework-%FRAMEWORK_NAME_1%-1.10607.0.tar.gz",
"sha1": "adce2cd30a830d71cb6572575bf08461b7b73c07",
"version": "1.10607.0",
"system": "*"
}
]
},
"frameworks": {
"%FRAMEWORK_NAME_1%": {
"package": "framework-%FRAMEWORK_NAME_1%",
"script": "builder/frameworks/%FRAMEWORK_NAME_1%.py"
},
"%FRAMEWORK_NAME_N%": {
"package": "framework-%FRAMEWORK_NAME_N%",
"script": "builder/frameworks/%FRAMEWORK_NAME_N%.py"
}
},
"packages": {
"toolchain-gccarmnoneabi": {
"type": "toolchain",
"version": ">=1.40803.0,<1.40805.0"
},
"framework-%FRAMEWORK_NAME_1%": {
"type": "framework",
"optional": true,
"version": "~1.10607.0"
},
"framework-%FRAMEWORK_NAME_N%": {
"type": "framework",
"optional": true,
"version": "~1.117.0"
},
"tool-direct-vcs-url": {
"type": "uploader",
"optional": true,
"version": "https://github.com/user/repo.git"
}
}
}

```

(continues on next page)

(continued from previous page)

}

Build Script `main.py`

Platform's build script is based on a next-generation build tool named `SCons`. PlatformIO has own built-in firmware builder `env.BuildProgram` with the deep libraries search. Please look into a base template of `main.py`.

```
"""
Build script for test.py
test-builder.py
"""

from os.path import join
from SCons.Script import AlwaysBuild, Builder, Default, DefaultEnvironment

env = DefaultEnvironment()

# A full list with the available variables
# http://www.scons.org/doc/production/HTML/scons-user.html#app-variables
env.Replace(
    AR="ar",
    AS="gcc",
    CC="gcc",
    CXX="g++",
    OBJCOPY="objcopy",
    RANLIB="ranlib",

    ARFLAGS=[ "... "],
    ASFLAGS=["flag1", "flag2", "flagN"],
    CCFLAGS=["flag1", "flag2", "flagN"],
    CXXFLAGS=["flag1", "flag2", "flagN"],
    LINKFLAGS=["flag1", "flag2", "flagN"],

    CPPDEFINES=[ "DEFINE_1", "DEFINE=2", "DEFINE_N" ],
    LIBS=[ "additional", "libs", "here" ],
    UPLOADER=join("$PIOPACKAGES_DIR", "tool-bar", "uploader"),
    UPLOADCMD="$UPLOADER $SOURCES"
)

env.Append(
    BUILDERS=dict(
        ElfToBin=Builder(
            action=" ".join([
                "$OBJCOPY",
                "-O",
                "binary",
                "$SOURCES",
                "$TARGET"]),
            suffix=".bin"
        )
    )
)
```

(continues on next page)

(continued from previous page)

```

# The source code of "platformio-build-tool" is here
# https://github.com/platformio/platformio-core/blob/develop/platformio/builder/tools/
#>platformio.py

#
# Target: Build executable and linkable firmware
#
target_elf = env.BuildProgram()

#
# Target: Build the .bin file
#
target_bin = env.ElfToBin(join("$BUILD_DIR", "firmware"), target_elf)

#
# Target: Upload firmware
#
upload = env.Alias(["upload"], target_bin, "$UPLOADCMD")
AlwaysBuild(upload)

#
# Target: Define targets
#
Default(target_bin)

```

Installation

1. Create `platforms` directory in `home_dir` if it doesn't exist.
2. Create `myplatform` directory in `platforms`
3. Copy `platform.json` and `builder/main.py` files to `myplatform` directory.
4. Search available platforms via `platformio platform search` command. You should see `myplatform` platform.
5. Install `myplatform` platform via `platformio platform install` command.

Now, you can use `myplatform` for the `platform` option in *Project Configuration File `platformio.ini`*.

Examples

Please take a look at the source code of PlatformIO Development Platforms.

1.13.2 Custom Embedded Board

PlatformIO has pre-built settings for the most popular embedded boards. This list is available:

- [Embedded Boards Explorer \(Web\)](#)
- [`platformio boards` \(CLI command\)](#)

Nevertheless, PlatformIO allows to create own board or override existing board's settings. All data is declared using `JSON-style` via `associative array` (name/value pairs).

Contents

- *Custom Embedded Board*
 - *JSON Structure*
 - *Installation*
 - *Examples*

JSON Structure

The key fields:

- build data will be used by *Development Platforms* and *Frameworks* builders
- frameworks is the list with supported *Frameworks*
- platform name of *Development Platforms*
- upload upload settings which depend on the platform

```
{  
    "build": {  
        "extra_flags": "-DHELLO_PLATFORMIO",  
        "f_cpu": "16000000L",  
        "hwids": [  
            [  
                "0x1234",  
                "0x0013"  
            ],  
            [  
                "0x4567",  
                "0x0013"  
            ]  
        ],  
        "mcu": "%MCU_TYPE_HERE%"  
    },  
    "frameworks": ["%LIST_WITH_SUPPORTED_FRAMEWORKS%"],  
    "name": "My Test Board",  
    "upload": {  
        "maximum_ram_size": 2048,  
        "maximum_size": 32256  
    },  
    "url": "http://example.com",  
    "vendor": "MyCompany"  
}
```

Installation

1. Create boards directory in *home_dir* if it doesn't exist.
2. Create myboard.json file in this boards directory.
3. Search available boards via *platformio boards* command. You should see myboard board.

Now, you can use myboard for the *board* option in *Project Configuration File platformio.ini*.

Note: You can have custom boards per project. In this case, please put your board's JSON files to `boards_dir`.

Examples

Please take a look at the source code of PlatformIO Development Platforms and navigate to `boards` folder of the repository.

1.14 PIO Account

Having **PIO Account** allows you to use extra professional features from **PIO Plus**:

- *PIO Unified Debugger*
- *PIO Remote*
- *PIO Unit Testing*
- Integration with *Cloud IDE*

A registration is **FREE**. No Credit Card Required.

1.14.1 PlatformIO IDE

PlatformIO IDE has built-in UI in PIO Home to manage PIO Account. You can crate a new account, reset a password or fetch authentication token.



PlatformIO Account

Having [PIO Account](#) allows you to use extra professional features:

- [PIO Remote™](#)
- [PIO Unified Debugger](#)
- [PIO Unit Testing](#)
- [Integration with Cloud IDEs](#)

[Log in](#)[Forgot Password?](#)

Need an Account? [Create a new one.](#)

1.14.2 User Guide (CLI)

1.15 PIO Remote

Your devices are always with you!

New in version 3.2: ([PIO Plus](#))

PIO Remote allows you to work remotely with devices from *Anywhere In The World*. No matter where are you now! Run a small and cross-platform [PIO Remote Agent](#) on a remote machine and you are able to list active devices (wireless + wired), to upload firmware (program), to process remote unit tests, or to start remote debugging session via **Remote Serial Port Monitor**.

Using PIO Remote you can share your devices with colleagues across your organization or friends. In combination with [Cloud IDE](#), you can create awesome things at any time when inspiration comes to you.

You should have [PIO Account](#) to work with **PIO Remote**. A registration is **FREE**.

Contents

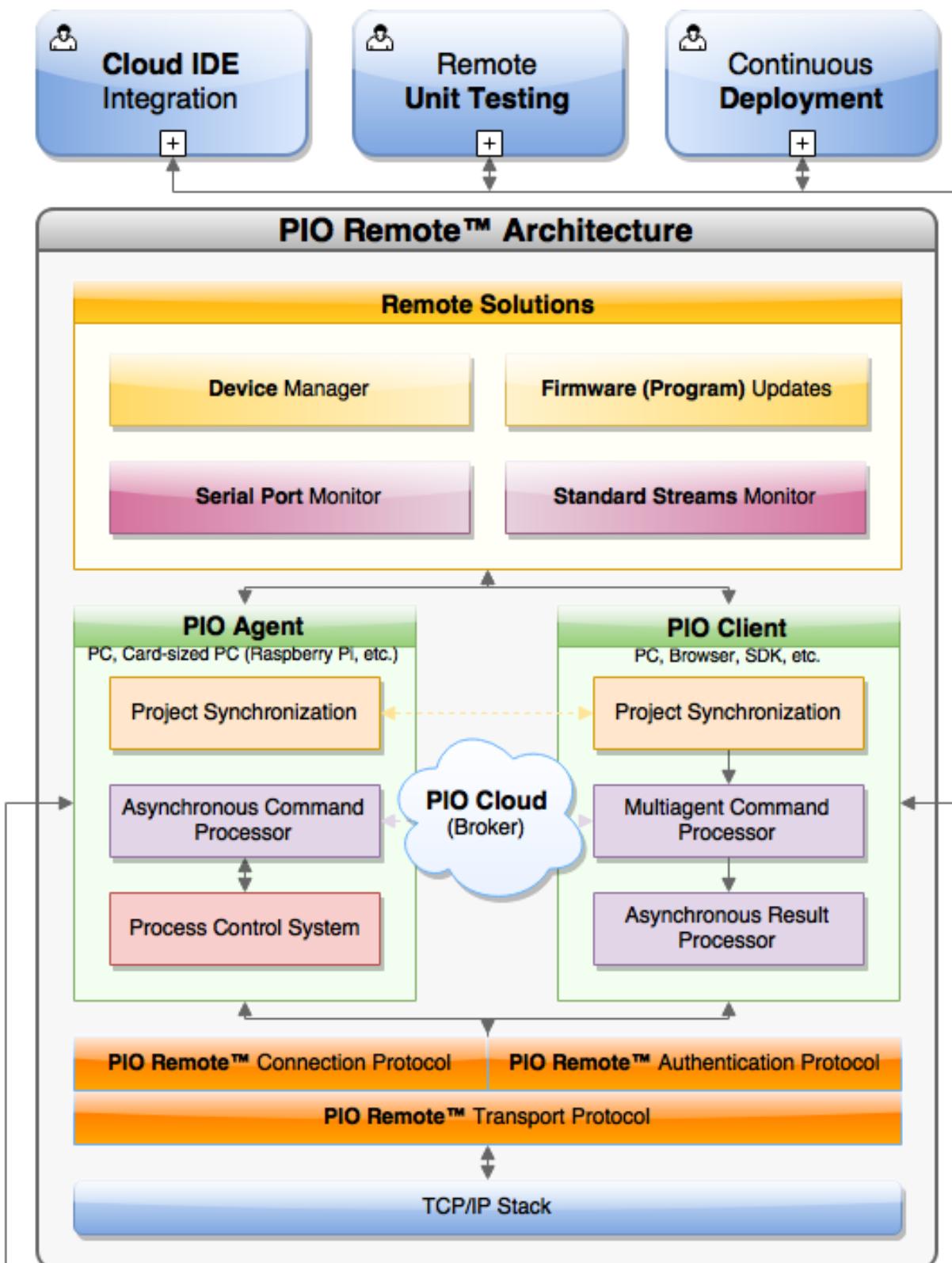
- [PIO Remote](#)

- *Features*
- *Technology*
- *Installation*
- *Quick Start*
- *User Guide (CLI)*

1.15.1 Features

- *Cloud IDE*
- *Remote Device Manager*
- *Remote Serial Port Monitor*
- *Remote Firmware Updates*
- **PIO Remote Share**
- Continuous Deployment
- Continuous Delivery
- Remote Unit Testing

1.15.2 Technology



USB Link

Ethernet Link

Wi-Fi Link

PIO Remote is an own PIO Plus technology for remote solutions without external dependencies to operation system or its software based on [client-server architecture](#). The Server component (PlatformIO Cloud) plays a role of coupling link between [PIO Remote Agent](#) and Client ([platformio remote](#), [Cloud IDE](#), [Continuous Integration](#), SDKs, etc.). When you start [PIO Remote Agent](#), it connects over the Internet with PlatformIO Cloud and listen for the actions/commands which you can send in Client role from anywhere in the world.

PIO Remote is multi-agents and multi-clients system. A single agent can be shared with multiple clients, where different clients can use the same agent. This approach allows to work with distributed hardware located in the different places, networks, etc.

This technology allows to work with remote devices in generic form as you do that with local devices using PlatformIO ecosystem. The only one difference is a prefix “remote” before each generic PlatformIO command. For example, listing of local and remote devices will look like [platformio device list](#) and [platformio remote device list](#).

1.15.3 Installation

PIO Remote is built into [PlatformIO IDE](#). Please open PlatformIO IDE Terminal and run `pio remote --help` command for usage (see [platformio remote](#)).

If you do not have [PlatformIO IDE](#), or use [Cloud IDE](#) or a card-sized PC (Raspberry Pi, BeagleBoard, etc.), please install [PlatformIO Core](#).

1.15.4 Quick Start

1. Start **PIO Remote Agent** using [platformio remote agent start](#) command on a **remote machine** where devices are connected physically or are accessible via network. **PIO Remote Agent works on Windows, macOS, Linux and Linux ARMv6+**. It means that you can use desktop machine, laptop or credit card sized PC (Raspberry Pi, BeagleBoard, etc.).

You can share own devices/hardware with friends, team or other developers using [platformio remote agent start --share](#) option.

2. Using **host machine** ([platformio remote](#), [Cloud IDE](#) Terminal in a browser, SDKs, etc.), please authorize via [platformio account login](#) command with the same credentials that you used on the previous step. Now, you can use [platformio remote](#) commands to work with **remote machine** and its devices.

You don't need to have networking or other access to **remote machine** where **PIO Remote** Agent is started.

If you use **PIO Remote** in pair with [Continuous Integration](#) or want automatically authorize, please set `PLATFORMIO_AUTH_TOKEN` system environment variable instead of using [platformio account login](#) command.

Note: In case with [Cloud IDE](#), your browser with Cloud IDE's VM is a “host machine”. The machine where devices are connected physically (your real PC) is called “remote machine” in this case. You should run **PIO Remote** Agent here (not in Cloud IDE's Terminal).

Note: Please use local IP as “upload port” when device is not connected directly to a remote machine where **PIO Remote** Agent is started but supports natively Over-the-Air (OTA) updates. For example, [Espresif 8266](#) and [Over-the-Air \(OTA\) update](#). In this case, the final command for remote OTA update will look as `platformio remote run -t upload --upload-port 192.168.0.255` or `platformio remote run -t upload --upload-port myesp8266.local`.

1.15.5 User Guide (CLI)

1.16 PIO Unified Debugger

It Simply Works. Easier than ever before!

New in version 3.4: ([PIO Plus](#))

Note: [Demo](#), discussions, request a support for new hardware.

- “1-click” solution, zero configuration
- Support for 200+ embedded boards (see below)
- Multiple architectures and development platforms
- Windows, MacOS, Linux (+ARMv6-8)
- Built-in into [PlatformIO IDE for Atom](#) and [PlatformIO IDE for VSCode](#)
- Integration with [Eclipse](#) and [Sublime Text](#)

You should have [PIO Account](#) to work with **PIO Unified Debugger**. A registration is **FREE**.

Hint: In our experience, [PlatformIO IDE for VSCode](#) has the best system performance, modern interface for **PIO Unified Debugger**, and users have found it easier to get started. Key debugging features of [PlatformIO IDE for VSCode](#):

- Conditional Breakpoints
 - Expressions and Watchpoints
 - Generic Registers
 - Peripheral Registers
 - Memory Viewer
 - Disassembly
 - Multi-thread support
 - A hot restart of an active debugging session
-

The screenshot shows the PlatformIO IDE interface with several panes:

- Left Sidebar:** Includes sections for VARIABLES, WATCH, BREAKPOINTS, PERIPHERALS, and REGISTERS.
- Code Editor (WiFi.cpp):** Shows C++ code for WiFi provisioning logic, including variable declarations and function implementations.
- Memory Dump:** Displays memory contents at address 0x42000800+1.
- Assembly View:** Shows the assembly code for WiFiClass::startProvision().
- Terminal:** Displays the output of the "platformio device monitor" command, showing the Miniterm setup and provision mode configuration.
- Bottom Status Bar:** Shows build status (2: Task - Monitor), file paths, and other system information.

Contents

- *Tutorials*
- *Configuration*
- *Tools & Debug Probes*
- *User Guide (CLI)*
- *Platforms*
- *Frameworks*
- *Boards*

1.16.1 Tutorials

- ThingForward: First steps with PlatformIO’s Unified Debugger
- [VIDEO] ThingForward - Intro to PIO Unified Debugger using ARM mbed OS and PlatformIO IDE for VSCode
- *STM32Cube HAL and Nucleo-F401RE: debugging and unit testing*

1.16.2 Configuration

Warning: For the JTAG probes implemented as USB devices (actually most of them), you need to configure the UDEV subsystem:

Linux Users: Install “udev” rules [99-platformio-udev.rules](#)

Windows Users: Please check that you have a correctly installed USB driver from board manufacturer

PIO Unified Debugger can be configured from *Project Configuration File platformio.ini*:

- *debug_tool*
- *debug_init_break*
- *debug_init_cmds*
- *debug_extra_cmds* (conditional project breakpoints, extra configuration, etc.)
- *debug_load_cmd*
- *debug_load_mode*
- *debug_server*
- *debug_port*
- *debug_svd_path*

1.16.3 Tools & Debug Probes

You can switch between debugging tools using *debug_tool* option.

Warning: You will need to install debug tool drivers depending on your system:

Windows Please check official documentation for your debug tool and install required drivers or use related tools, such as Zadig.

Mac You don’t need to install extra drivers. Nevertheless, please check official documentation.

Linux Please install “udev” rules [99-platformio-udev.rules](#)

- *Atmel-ICE*
- *Black Magic Probe*
- *CMSIS-DAP*
- *FTDI Chip*

- *J-LINK*
- *Mini-Module FT2232H*
- *MSP Debug*
- *Olimex ARM-USB-OCD*
- *Olimex ARM-USB-OCD-H*
- *Olimex ARM-USB-TINY*
- *Olimex ARM-USB-TINY-H*
- *TI-ICDI*
- *ST-LINK*
- *Custom*
 - *Examples*
 - * *Black Magic Probe*
 - * *J-Link and ST Nucleo*
 - * *J-Link as debugger and uploader*
 - * *ST-Util and ST-Link*
 - * *OpenOCD and ST-Link*
 - * *pyOCD and CMSIS-DAP*

Atmel-ICE

Configuration `debug_tool = atmel-ice`



Picture

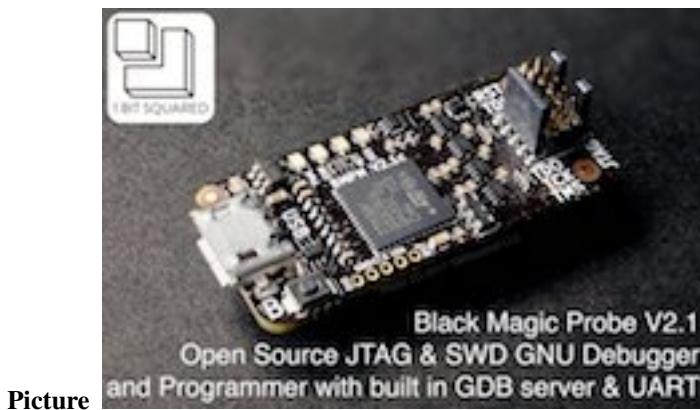
Description Atmel-ICE is a powerful development tool for debugging and programming ARM® Cortex®-M based SAM and AVR microcontrollers with on-chip debug capability. [Vendor information...](#)

Compatible Platforms

- *Atmel SAM*

Black Magic Probe

Configuration `debug_tool = blackmagic`



Picture

Description The Black Magic Probe is a modern, in-application debugging tool for embedded microprocessors. It is able to control and examine the state of the target microprocessor using a JTAG or Serial Wire Debugging (SWD) port and on-chip debug logic provided by the microprocessor. The probe connects to a host computer using a standard USB interface. [Vendor information...](#)

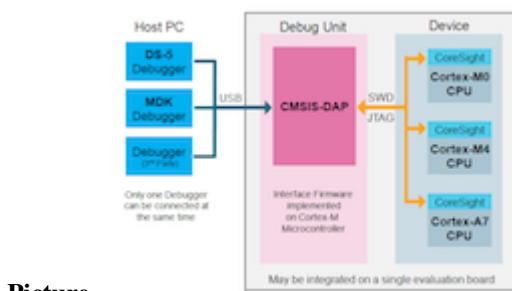
Also, see [Custom](#) debugging configuration with Black Magic Probe.

Compatible Platforms

- [Atmel SAM](#)
- [Freescale Kinetis](#)
- [Nordic nRF51](#)
- [Nordic nRF52](#)
- [NXP LPC](#)
- [Silicon Labs EFM32](#)
- [ST STM32](#)

CMSIS-DAP

Configuration `debug_tool = cmsis-dap`



Picture

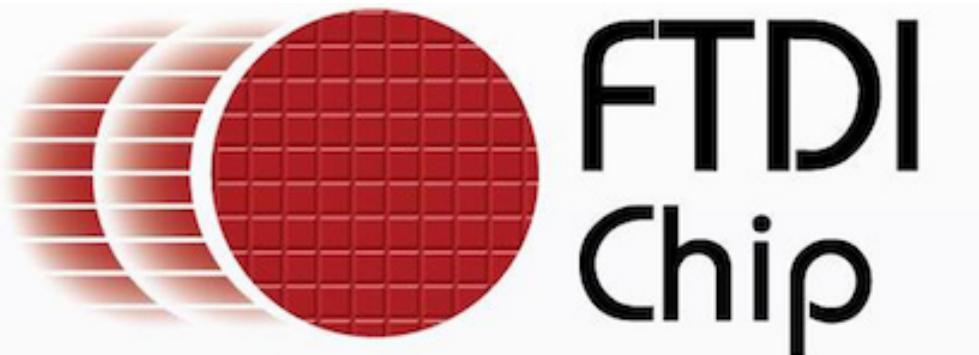
Description CMSIS-DAP is generally implemented as an on-board interface chip, providing direct USB connection from a development board to a debugger running on a host computer on one side, and over JTAG (Joint Test Action Group) or SWD (Serial Wire Debug) to the target device to access the Coresight DAP on the other. [Vendor information...](#)

Compatible Platforms

- *Atmel SAM*
- *Freescale Kinetis*
- *Maxim 32*
- *Nordic nRF51*
- *Nordic nRF52*
- *NXP LPC*
- *ST STM32*
- *WIZNet W7500*

FTDI Chip

Configuration `debug_tool = ftdi`



Picture

Description FTDI Chip develops innovative silicon solutions that enhance interaction with today's technology. When a designer needs to add a USB port, rest assured that FTDI Chip has a full range of USB solutions to get the job done... <http://www.ftdichip.com/USB.html?utm_source=platformio&utm_medium=docs>__

Drivers

Windows See <https://community.platformio.org/t/esp32-pio-unified-debugger/4541/20>

Mac You don't need to install extra drivers. Nevertheless, please check official documentation.

Linux Please install “udev” rules [99-platformio-udev.rules](#).

Compatible Platforms

- *Espressif 32*
- *RISC-V*
- *Samsung ARTIK*

J-LINK

Configuration `debug_tool = jlink`



Picture

Description SEGGER J-Links are the most widely used line of debug probes available today. They've proven their value for more than 10 years with over 400,000 units sold, including OEM versions and on-board solutions. This popularity stems from the unparalleled performance, extensive feature set, large number of supported CPUs, and compatibility with all popular development environments.
[Vendor information...](#)

- [Install J-Link GDB Server](#)
- [J-Link Supported Devices](#)

Also, see [Custom](#) debugging configuration with J-Link GDB Server.

Compatible Platforms

- [Atmel SAM](#)
- [Freescale Kinetis](#)
- [Infineon XMC](#)
- [Nordic nRF51](#)
- [Nordic nRF52](#)
- [NXP LPC](#)
- [Silicon Labs EFM32](#)
- [ST STM32](#)
- [Teensy](#)
- [WIZNet W7500](#)

Mini-Module FT2232H

Configuration `debug_tool = minimodule`



Picture

Description The FT2232H Mini Module is a USB to dual channel serial/MPSSE/FIFO interface converter module based on the FT2232H USB Hi-Speed IC. The FT2232H handles all the USB signalling and protocol handling. The module provides access to device I/O interfaces via 2 double row 0.1" pitch male connectors. The module is ideal for development purposes to quickly prove functionality of adding USB to a target design. [Vendor information...](#)

Wiring Connections

FT2232H Mini-Module Pin	Board JTAG Pin
AD0 [CN2-7]	TCK
AD1 [CN2-10]	TDI
AD2 [CN2-9]	TDO
AD3 [CN2-12]	TMS
AC2 [CN2-20]	RST
GND [CN2-2]	GND

You will also need to connect Vbus [CN3-1] to Vcc [CN3-3] of FT2232H Mini-Module to power the FTDI chip. See [FT2232H Mini-Module Datasheet](#)

Drivers

Windows See <https://community.platformio.org/t/esp32-pio-unified-debugger/4541/20>

Mac You don't need to install extra drivers. Nevertheless, please check official documentation.

Linux Please install "udev" rules [99-platformio-udev.rules](#).

Compatible Platforms

- [Espressif 32](#)

MSP Debug

Configuration `debug_tool = mspdebug`

Description The MSP debug stack (MSPDS) for all MSP430™ microcontrollers (MCUs) and SimpleLink™ MSP432™ devices consists of a static library on the host system side as well as an embedded firmware that runs on debug tools including the MSP-FET, MSP-FET430UIF or on-board eZ debuggers. It is the bridging element between all PC software and all MSP430 and SimpleLink

MSP432 microcontroller derivatives and handles tasks such as code download, stepping through code or break points. [Vendor information...](#)

Compatible Platforms

- [TI MSP430](#)

Olimex ARM-USB-OCD

Configuration `debug_tool = olimex-arm-usb-ocd`



Picture

Description 3-IN-1 fast USB ARM/ESP32 JTAG, USB-to-RS232 virtual port and power supply 5-9-12VDC device (supported by OpenOCD ARM debugger software). [Vendor information...](#)

Wiring Connections

Olimex ARM-USB-OCD Pin	Board JTAG Pin
11	TCK
5	TDI
13	TDO
7	TMS
3	RST
4	GND

Drivers

Windows See <https://community.platformio.org/t/esp32-pio-unified-debugger/4541/20>

Mac You don't need to install extra drivers. Nevertheless, please check official documentation.

Linux Please install “udev” rules [99-platformio-udev.rules](#).

Compatible Platforms

- [Espressif 32](#)

Olimex ARM-USB-OCD-H

Configuration `debug_tool = olimex-arm-usb-ocd-h`



Picture

Description High-speed 3-IN-1 fast USB ARM/ESP32 JTAG, USB-to-RS232 virtual port and power supply 5VDC device. [Vendor information...](#)

Wiring Connections

Olimex ARM-USB-OCD-H Pin	Board JTAG Pin
11	TCK
5	TDI
13	TDO
7	TMS
3	RST
4	GND

Drivers

Windows See <https://community.platformio.org/t/esp32-pio-unified-debugger/4541/20>

Mac You don't need to install extra drivers. Nevertheless, please check official documentation.

Linux Please install “udev” rules [99-platformio-udev.rules](#).

Compatible Platforms

- [Espressif 32](#)

Olimex ARM-USB-TINY

Configuration `debug_tool = olimex-jtag-tiny`



Picture

Description Low-cost and high-speed ARM/ESP32 USB JTAG. [Vendor information...](#)

Wiring Connections

Olimex ARM-USB-TINY Pin	Board JTAG Pin
11	TCK
5	TDI
13	TDO
7	TMS
3	RST
4	GND

Drivers

Windows See <https://community.platformio.org/t/esp32-pio-unified-debugger/4541/20>

Mac You don't need to install extra drivers. Nevertheless, please check official documentation.

Linux Please install “udev” rules [99-platformio-udev.rules](#).

Compatible Platforms

- [Espressif 32](#)

Olimex ARM-USB-TINY-H

Configuration `debug_tool = olimex-arm-usb-tiny-h`



Picture

Description Low-cost and high-speed ARM/ESP32 USB JTAG. [Vendor information...](#)

Wiring Connections

Olimex ARM-USB-TINY-H Pin	Board JTAG Pin
11	TCK
5	TDI
13	TDO
7	TMS
3	RST
4	GND

Drivers

Windows See <https://community.platformio.org/t/esp32-pio-unified-debugger/4541/20>

Mac You don't need to install extra drivers. Nevertheless, please check official documentation.

Linux Please install "udev" rules [99-platformio-udev.rules](#).

Compatible Platforms

- [Espressif 32](#)
- [RISC-V](#)

TI-ICDI

Configuration `debug_tool = ti-icdi`

Description Tiva™ C Series evaluation and reference design kits provide an integrated In-Circuit Debug Interface (ICDI) which allows programming and debugging of the onboard C Series microcontroller. [Vendor information...](#)

Compatible Platforms

- [TI TIVA](#)

ST-LINK

Configuration `debug_tool = stlink`



Picture

Description The ST-LINK is an in-circuit debugger and programmer for the STM8 and STM32 microcontroller families. The single wire interface module (SWIM) and JTAG/serial wire debugging (SWD) interfaces are used to communicate with any STM8 or STM32 microcontroller located on an application board. [Vendor information...](#)

Compatible Platforms

- [Nordic nRF51](#)
- [Nordic nRF52](#)
- [ST STM32](#)

Custom

Configuration `debug_tool = custom`

Custom debugging configuration

- `debug_init_break`
- `debug_init_cmds`
- `debug_extra_cmds` (conditional project breakpoints, extra configuration, etc.)
- `debug_load_cmd`
- `debug_server`
- `debug_port`

Examples

- *Black Magic Probe*
- *J-Link and ST Nucleo*
- *J-Link as debugger and uploader*
- *ST-Util and ST-Link*
- *OpenOCD and ST-Link*
- *pyOCD and CMSIS-DAP*

Black Magic Probe

Black Magic Probe with a custom `debug_port` (list ports with `platformio device list`)

```
[env:debug]
platform = ...
board = ...
framework = ...
debug_tool = custom
; set here a valid port...
debug_port = /dev/cu.usbmodem7BB07991
debug_init_cmds =
    target extended-remote $DEBUG_PORT
    monitor swdp_scan
    attach 1
    set mem inaccessible-by-default off
$INIT_BREAK
$LOAD_CMD
```

J-Link and ST Nucleo

Segger J-Link probe and ST Nucleo F446RE board in pair with J-Link GDB Server:

- Install J-Link GDB Server

- Convert ST-LINK On-Board Into a J-Link

Note: You can use configuration below in pair with other boards, not only with ST Nucleo F446RE. In this case, please replace STM32F446RE with your own device name in `debug_server` option.

See full list with [J-Link Supported Devices](#).

```
[env:debug_jlink]
platform = ststm32
framework = mbed
board = nucleo_f446re
debug_tool = custom
debug_server =
    /full/path/to/JLinkGDBServerCL
    -singlerun
    -if
    SWD
    -select
    USB
    -port
    2331
    -device
    STM32F446RE
```

J-Link as debugger and uploader

Segger J-Link probe as debugger and uploader for a custom Teensy-based board. If you plan to use with other board, please change device MK20DX256xxxx7 to a valid identifier. See supported J-Link devices at [J-LINK](#).

- Install J-Link GDB Server

```
[env:jlink_debug_and_upload]
platform = teensy
framework = arduino
board = teensy31
extra_scripts = extra_script.py
upload_protocol = custom
debug_tool = custom
debug_server =
    /full/path/to/JLinkGDBServerCL
    -singlerun
    -if
    SWD
    -select
    USB
    -port
    2331
    -device
    MK20DX256xxxx7
```

`extra_script.py`

Place this file on the same level as *Project Configuration File* `platformio.ini`.

```

from os import makedirs
from os.path import isdir, join
Import('env')

# Optional block, only for Teensy
env.AddPostAction(
    "$BUILD_DIR/firmware.hex",
    env.VerboseAction(" ".join([
        "sed", "-i.bak",
        "<",
        "s/:10040000FFFFFFFFFFFFFFDEF9FFF23/:10040000FFFFFFFFFFFFFFFFFEEFFF/",
        ">",
        "$BUILD_DIR/firmware.hex"
    ]), "Fixing $BUILD_DIR/firmware.hex secure flash flags"))

def _jlink_cmd_script(env, source):
    build_dir = env.subst("$BUILD_DIR")
    if not isdir(build_dir):
        makedirs(build_dir)
    script_path = join(build_dir, "upload.jlink")
    commands = ["h", "loadbin %s,0x0" % source, "r", "q"]
    with open(script_path, "w") as fp:
        fp.write("\n".join(commands))
    return script_path

env.Replace(
    __jlink_cmd_script=_jlink_cmd_script,
    UPLOADER="/full/path/to/JLink",
    UPLOADERFLAGS=[
        "-device", "MK20DX256xxx7",
        "-speed", "4000",
        "-if", "swd",
        "-autoconnect", "1"
    ],
    UPLOADCMD=' "$UPLOADER" $UPLOADERFLAGS -CommanderScript ${__jlink_cmd_script(__env__, SOURCE)} '
)

```

ST-Util and ST-Link

On-board ST-Link V2/V2-1 in pair with ST-Util GDB Server:

```

[env:debug]
platform = ststm32
framework = mbed
board = ...
debug_tool = custom
debug_port = :4242
debug_server = ${PLATFORMIO_HOME_DIR}/packages/tool-stlink/bin/st-util

```

OpenOCD and ST-Link

On-board ST-Link V2/V2-1 in pair with OpenOCD GDB Server:

```
[env:debug]
platform = ststm32
framework = mbed
board = ...
debug_tool = custom
debug_server =
    $PLATFORMIO_HOME_DIR/packages/tool-openocd/bin/openocd
    -f
    $PLATFORMIO_HOME_DIR/packages/tool-openocd/scripts/board/st_nucleo_f4.cfg
```

pyOCD and CMSIS-DAP

Using pyOCD for CMSIS-DAP based boards

Firstly, please install [pyOCD](#) and check that `pyocd-gdbserver --version` command works.

```
[env:debug]
platform = ...
board = ...
framework = mbed
debug_tool = custom
debug_server = pyocd-gdbserver
```


1.16.4 User Guide (CLI)

1.16.5 Platforms

Name	Description
Atmel SAM	Atmel SMART offers Flash- based ARM products based on the ARM Cortex-M0+, Cortex-M3 and Cortex-M4 architectures, ranging from 8KB to 2MB of Flash including a rich peripheral and feature mix.
Espressif 32	Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.
Freescale Kinetis	Freescale Kinetis Microcontrollers is family of multiple hardware- and software-compatible ARM Cortex-M0+, Cortex-M4 and Cortex-M7-based MCU series. Kinetis MCUs offer exceptional low-power performance, scalability and feature integration.
Infineon XMC	Infineon has designed the XMC microcontrollers for real-time critical applications with an industry-standard core. The XMC microcontrollers can be integrated with the Arduino platform
Maxim 32	Maxim's microcontrollers provide low-power, efficient, and secure solutions for challenging embedded applications. Maxim's processors embed cutting-edge technologies to secure data and intellectual property, proven analog circuitry for real-world applications, and battery-conserving low power operation.
Nordic nRF51	The Nordic nRF51 Series is a family of highly flexible, multi-protocol, system-on-chip (SoC) devices for ultra-low power wireless applications. nRF51 Series devices support a range of protocol stacks including Bluetooth Smart (previously called Bluetooth low energy), ANT and proprietary 2.4GHz protocols such as Gazell.
Nordic nRF52	The nRF52 Series are built for speed to carry out increasingly complex tasks in the shortest possible time and return to sleep, conserving precious battery power. They have a Cortex-M4F processor and are the most capable Bluetooth Smart SoCs on the market.
NXP LPC	The NXP LPC is a family of 32-bit microcontroller integrated circuits by NXP Semiconductors. The LPC chips are grouped into related series that are based around the same 32-bit ARM processor core, such as the Cortex-M4F, Cortex-M3, Cortex-M0+, or Cortex-M0. Internally, each microcontroller consists of the processor core, static RAM memory, flash memory, debugging interface, and various peripherals.
RISC-V	RISC-V is an open, free ISA enabling a new era of processor innovation through open standard collaboration. Born in academia and research, RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.
Samsung ARTIK	The Samsung ARTIK Smart IoT platform brings hardware modules and cloud services together, with built-in security and an ecosystem of tools and partners to speed up your time-to-market.
Silicon Labs EFM32	Silicon Labs EFM32 Gecko 32-bit microcontroller (MCU) family includes devices that offer flash memory configurations up to 256 kB, 32 kB of RAM and CPU speeds up to 48 MHz. Based on the powerful ARM Cortex-M core, the Gecko family features innovative low energy techniques, short wake-up time from energy saving modes and a wide selection of peripherals, making it ideal for battery operated applications and other systems requiring high performance and low-energy consumption.
ST STM32	The STM32 family of 32-bit Flash MCUs based on the ARM Cortex-M processor is designed to offer new degrees of freedom to MCU users. It offers a 32-bit product range that combines very high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.
Teensy	Teensy is a complete USB-based microcontroller development system, in a very small footprint, capable of implementing many types of projects. All programming is done via the USB port. No special programmer is needed, only a standard USB cable and a PC or Macintosh with a USB port.
TI MSP430	MSP430 microcontrollers (MCUs) from Texas Instruments (TI) are 16-bit, RISC-based, mixed-signal processors designed for ultra-low power. These MCUs offer the lowest power consumption and the perfect mix of integrated peripherals for thousands of applications.
TI TIVA 508	Texas Instruments TM4C12x MCUs offer the industry's most popular ARM Cortex-M4 core with scalable memory and package options, unparalleled connectivity peripherals, advanced application functions, industry-leading analog integration, and extensive software solutions.
WIZ-Net W7500	The IOP (Internet Offload Processor) W7500 is the one-chip solution which integrates an ARM Cortex-M0, 128KB Flash and hardwired TCP/IP core for various embedded application platforms especially requiring Internet of things

1.16.6 Frameworks

Name	Description
Arduin	Arduino Wiring-based Framework allows writing cross-platform software to control devices attached to a wide range of Arduino boards to create all kinds of creative coding, interactive objects, spaces or physical experiences.
CM-SIS	The ARM Cortex Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for the Cortex-M processor series and specifies debugger interfaces. The CMSIS enables consistent and simple software interfaces to the processor for interface peripherals, real-time operating systems, and middleware. It simplifies software re-use, reducing the learning curve for new microcontroller developers and cutting the time-to-market for devices.
Energia	Energia Wiring-based framework enables pretty much anyone to start easily creating microcontroller-based projects and applications. Its easy-to-use libraries and functions provide developers of all experience levels to start blinking LEDs, buzzing buzzers and sensing sensors more quickly than ever before.
ESP-IDF	Espressif IoT Development Framework. Official development framework for ESP32.
Freedom E SDK	Open Source Software for Developing on the SiFive Freedom E Platform
libOpenCortex	The libOpenCM3 framework aims to create a free/libre/open-source firmware library for various ARM Cortex-M0(+)/M3/M4 microcontrollers, including ST STM32, TI Tiva and Stellaris, NXP LPC 11xx, 13xx, 15xx, 17xx parts, Atmel SAM3, Energy Micro EFM32 and others.
mbed	The mbed framework The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.
Simba	Simba is an RTOS and build framework. It aims to make embedded programming easy and portable.
SPL	The ST Standard Peripheral Library provides a set of functions for handling the peripherals on the STM32 Cortex-M3 family. The idea is to save the user (the new user, in particular) having to deal directly with the registers.
STM32Cube	STM32Cube embedded software libraries, including: The HAL hardware abstraction layer, enabling portability between different STM32 devices via standardized API calls; The Low-Layer (LL) APIs, a light-weight, optimized, expert oriented set of APIs designed for both performance and runtime efficiency.
Tizen RT	Tizen RT is a lightweight RTOS-based platform to support low-end IoT devices

1.16.7 Boards

Note: For more detailed board information please scroll tables below by horizontal.

1BitSquared

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
1bitsy_stm32f4151Bitsy	1Bitsy	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F415RG	16MHz	1MB	128KB

96Boards

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
b96b_f446	96Boards B96B-F446VE	ST STM32	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32F446VE	16MHz	512KB	128KB

Adafruit

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
adafruit_circuitplaygroundexpress	Adafruit Playground Circuit Playground Express	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD24	8MHz	256KB	32KB
adafruit_featherm0	Adafruit Feather M0	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD24	8MHz	256KB	32KB
adafruit_featherm0express	Adafruit Feather M0 Express	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD24	8MHz	256KB	32KB
adafruit_featherm4	Adafruit Feather M4 (SAMD51)	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD51	20MHz	196KB	192KB
adafruit_gemma	Adafruit Gemma M0	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD24	8MHz	256KB	32KB
adafruit_itsybitsy	Adafruit Itsy Bitsy M0	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD24	8MHz	256KB	32KB
adafruit_itsybitsym4	Adafruit Itsy Bitsy M4 (SAMD51)	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD51	20MHz	196KB	192KB
adafruit_metro	Adafruit Metro M0 Express	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD24	8MHz	256KB	32KB
adafruit_metrom4	Adafruit Metro M4 (SAMD51)	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD51	20MHz	196KB	192KB
adafruit_pirkey	Adafruit pIRkey	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD24	8MHz	256KB	32KB
adafruit_trinketm0	Adafruit Trinket M0	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD24	8MHz	256KB	32KB
featheresp32	Adafruit ESP32 Feather	Espressif 32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	240MHz	4MB	320KB

Aiyarafun

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
node32	Node32	Espressif 32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

Arduino

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
due	Arduino Due (Programming Port)	Atmel SAM	<i>Black Magic Probe, J-LINK</i>	AT91SAM	134MHz	512KB	32KB
dueUSB	Arduino Due (USB Native Port)	Atmel SAM	<i>Black Magic Probe, J-LINK</i>	AT91SAM	134MHz	512KB	32KB
mkr1000USB	Arduino MKR1000	Atmel SAM	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB
mkrfox1200	Arduino MKR FOX 1200	Atmel SAM	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB
mkrgsm1400	Arduino MKR GSM 1400	Atmel SAM	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB
mkrwan1300	Arduino MKR WAN 1300	Atmel SAM	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB
mkrzero	Arduino MKRZERO	Atmel SAM	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB
mzeroUSB	Arduino M0	Atmel SAM	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB
mzeropro	Arduino M0 Pro (Programming/Debug Port)	Atmel SAM	<i>CMSIS-DAP (on-board), Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB
mzeroproUSB	Arduino M0 Pro (Native USB Port)	Atmel SAM	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB
tian	Arduino Tian	Atmel SAM	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB
zero	Arduino Zero (Programming/Debug Port)	Atmel SAM	<i>CMSIS-DAP (on-board), Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB
zeroUSB	Arduino Zero (USB Native Port)	Atmel SAM	<i>Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21	48MHz	256KB	32KB

Armstrap

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
armstrap_eagle1024	Armstrap Eagle 1024	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F417VIT ₆₈ MHz	1MHz	1MB	192KB
armstrap_eagle2048	Armstrap Eagle 2048	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F427VIT ₆₈ MHz	1.99MHz	256KB	
armstrap_eagle512	Armstrap Eagle 512	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F407VET ₆₈ MHz	512KB	192KB	

Atmel

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
samd21_xpro	Atmel SAMD21-XPRO	Atmel SAM	<i>CMSIS-DAP (on-board), Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21J188MHz	256KB	32KB	
samd21g18a	Atmel ATSAMW25-XPRO	Atmel SAM	<i>CMSIS-DAP (on-board), Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMD21G188MHz	256KB	32KB	
saml21_xpro	Atmel SAML21-XPRO-B	Atmel SAM	<i>CMSIS-DAP (on-board), Atmel-ICE, Black Magic Probe, J-LINK</i>	SAML21J188MHz	256KB	32KB	
samr21_xpro	Atmel ATSAMR21-XPRO	Atmel SAM	<i>CMSIS-DAP (on-board), Atmel-ICE, Black Magic Probe, J-LINK</i>	SAMR21G188MHz	256KB	32KB	

BBC

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
bbcmicrobit	BBC micro:bit	Nordic nRF51	<i>CMSIS-DAP (on-board)</i>	NRF51822	16MHz	256KB	16KB
bbcmicrobit_b	BBC micro:bit B(S130)	Nordic nRF51	<i>CMSIS-DAP (on-board)</i>	NRF51822	16MHz	256KB	16KB

BluzDK

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
bluz_dk	BluzDK	Nordic nRF51	<i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF51822	32MHz	256KB	32KB

CQ Publishing

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lpc11u35_500	CQ Publishing TG-LPC11U35-501	NXP LPC	<i>Black Magic Probe, J-LINK</i>	LPC11U3	548MHz	64KB	10KB

DFRobot

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
firebee	FireBee240 ESP32	Espresso32	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32	2240MHz	4MB	320KB

DOIT

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM	
esp32doit	DOIT DEVKIT V1	ESP32	ESPRESSO32	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32	2240MHz	4MB	320KB

Delta

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
delta_dfbm_nnq620	Delta DFBM-NQ620	Nordic nRF52	<i>CMSIS-DAP (on-board), Black Magic Probe, J-LINK, ST-LINK</i>	NRF52	32MHz	512KB	64KB
delta_dfcm_nnn50	Delta DFCM-NNN50	Nordic nRF51	<i>CMSIS-DAP (on-board)</i>	NRF51	22MHz	256KB	16KB
dfcm_nnn40	Delta DFCM-NNN40	Nordic nRF51	<i>CMSIS-DAP (on-board)</i>	NRF51	22MHz	256KB	32KB

Digistump

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
digix	Digistump DigiX	Atmel SAM	<i>Black Magic Probe, J-LINK</i>	AT91SAM3X8E	84MHz	512KB	28KB

Dongsen Technology

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
pocket	Dongsen Tech Pocket 32	Espressif 32	- Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

DycodeX

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp32	ESP32 Pectro32	Espressif 32	- Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

ESP32vn

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp32vn	ESP32vn IoT Uno	Espressif 32	- Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

Elektor Labs

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
elektor_cocor	CoCo-ri-Co!	NXP LPC	CMSIS-DAP (on-board), Black Magic Probe, J-LINK	LPC812	30MHz	16KB	4KB

Embedded Artists

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lpc11u35	EA LPC11U35 QuickStart Board	NXP LPC	Black Magic Probe, J-LINK	LPC11U35	48MHz	64KB	10KB
lpc4088	Embedded Artists LPC4088 QuickStart Board	NXP LPC	CMSIS-DAP (on-board), J-LINK	LPC4088	120MHz	512KB	96KB
lpc4088	Embedded Artists LPC4088 Display Module	NXP LPC	CMSIS-DAP (on-board), J-LINK	LPC4088	120MHz	512KB	96KB

Espotel

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
elmo_f411	Espotel LoRa Module	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i> (default)	STM32F411RET	100MHz	512KB	128KB

Espressif

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
esp-wroverkit	Espressif ESP-WROVER-KIT	Espressif 32	<i>Espresso-FTDI Chip</i> (default, on-board), <i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32	240MHz	4MB	320KB
esp32devkit	Espressif ESP32 Dev Module	Espressif 32	<i>Espresso-Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32	240MHz	4MB	320KB

Freescale

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
IBMEthernet	Ethernet IoT Starter Kit	Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), J-LINK	MK64FN1M0V100MHz	1MB	256KB
frdm_k20d	Freescale Kinetis FRDM-K20D50M	Freescale Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), J-LINK	MK20DX128V18MHz	128KB	16KB
frdm_k22f	Freescale Kinetis FRDM-K22F	Freescale Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), J-LINK	MK22FN512V100MHz	512KB	128KB
frdm_k64f	Freescale Kinetis FRDM-K64F	Freescale Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), J-LINK	MK64FN1M0V100MHz	1MB	256KB
frdm_k66f	Freescale Kinetis FRDM-K66F	Freescale Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), J-LINK	MK66FN2M0V100MHz	2MB	256KB
frdm_k105z	Freescale Kinetis FRDM-KL05Z	Freescale Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), J-LINK	MKL05Z32V18MHz	32KB	4KB
frdm_kl25z	Freescale Kinetis FRDM-KL25Z	Freescale Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), <i>Black Magic Probe, J-LINK</i>	MKL25Z128V18MHz	128KB	16KB
frdm_kl27z	Freescale Kinetis FRDM-KL27Z	Freescale Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), <i>Black Magic Probe, J-LINK</i>	MKL27Z64V18MHz	64KB	16KB
frdm_kl43z	Freescale Kinetis FRDM-KL43Z	Freescale Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), J-LINK	MKL43Z256V18MHz	256KB	32KB
frdm_kl46z	Freescale Kinetis FRDM-KL46Z	Freescale Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), J-LINK	MKL46Z256V18MHz	256KB	32KB
frdm_kw41z	Freescale Kinetis FRDM-KW41Z	Freescale Kinetis	<i>Freescale Kinetis</i>	CMSIS-DAP (on-board), J-LINK	MKW41Z512V18MHz	512KB	128KB

Generic

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
bluepill_f103B	BluePill F103C8	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	64KB	20KB	
genericSTM32F0103C8	(20k RAM. 64k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	64KB	20KB	
genericSTM32F0103CB	(20k RAM. 128k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CBMHz	128KB	20KB	
genericSTM32F0103R8	(20k RAM. 64 Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103R8MHz	64KB	20KB	
genericSTM32F0103RB	(20k RAM. 128k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103RBMHz	128KB	20KB	
genericSTM32F0103RC	(48k RAM. 256k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103RCMHz	256KB	48KB	
genericSTM32F0103RE	(64k RAM. 512k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103REMHz	512KB	64KB	
genericSTM32F0103VC	(48k RAM. 256k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103VCMHz	256KB	48KB	
genericSTM32F0103VE	(64k RAM. 512k Flash)	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103VENMHz	512KB	64KB	
genericSTM32F007VE	(192k RAM. 512k Flash)	ST STM32	ST-LINK	STM32F407VENMHz	502.23KB	28KB	

Hornbill

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
hornbill13	hornbill	Espressif ESP32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB
hornbill13	hornbillma	Espressif ESP32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

Infineon

ID	Name	Platform	Debug	MCU	Fre-quency	Flash	RAM
xmc1100_boot_kit	XMC1100 Boot Kit	Infineon XMC	J-LINK (on-board)	XMC110032MHz	64KB	64KB	
xmc1100_h_bridge	XMC1100 H-Bridge 2Go	Infineon XMC	J-LINK (on-board)	XMC110032MHz	64KB	64KB	
xmc1100_xmc2go	XMC1100 XMC2Go	Infineon XMC	J-LINK (on-board)	XMC110032MHz	64KB	64KB	
xmc1300_boot_kit	XMC1300 Boot Kit	Infineon XMC	J-LINK (on-board)	XMC130032MHz	64KB	64KB	
xmc1300_sense2go	XMC1300 Sense2GoL	Infineon XMC	J-LINK (on-board)	XMC130032MHz	64KB	122.23KB	
xmc4200_distance	XMC4200 Distance2Go	Infineon XMC	J-LINK (on-board)	XMC420080MHz	250KB	256KB	
xmc4700_relax_kit	XMC4700 Relax Kit	Infineon XMC	J-LINK (on-board)	XMC4700144MHz	2.00MB	1.95MB	

JKSoft

ID	Name	Platform	Debug	MCU	Fre-quency	Flash	RAM
wallbot_ble	JKSoft Wallbot BLE	Nordic nRF51	CMSIS-DAP (on-board)	NRF51822	16MHz	128KB	16KB

LeafLabs

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
maple	Maple	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103RETMHz	108KB	17KB	
maple_mini_b2	Maple Mini Bootloader 2.0	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CETMHz	120KB	20KB	
maple_mini_original	Maple Mini Original	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F103CETMHz	108KB	17KB	

MH-ET Live

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
mhetesp32m1Devkit	ET LIVE ESP32DevKit32	Espressif	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	
mhetesp32m1nikit	ET LIVE ESP32MiniKit32	Espressif	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32240MHz	4MB	320KB	

MXChip

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mxchip_az3166	Microsoft Azure IoT Development Kit (MXChip AZ3166)	ST STM32	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe</i> , <i>J-LINK</i>	STM32F412ZG	120MHz	1MB	256KB

Macchina

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
macchina2	Macchina M2	Atmel SAM	<i>Black Magic Probe</i> , <i>J-LINK</i>	AT91SAM3X8E	84MHz	512KB	32KB

Maxim

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
max32600m	Maxim ARM mbed Enabled Development Platform for MAX32600	Maxim 32	<i>CMSIS-DAP</i> (on-board)	MAX32600	104MHz	256KB	32KB
maxwsnenv	Maxim Wireless Sensor Node Demonstrator	Maxim 32	<i>CMSIS-DAP</i>	MAX32600	104MHz	256KB	32KB

Micromint

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lpc4330_m	Bambino-210E	NXP LPC	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe</i> , <i>J-LINK</i>	LPC4330	204MHz	8MB	264KB

MikroElektronika

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
hexiwear	Hexi-wear	Freescale Kinetis	<i>CMSIS-DAP</i> , <i>J-LINK</i>	MK64FN1M0VDC1	12120MHz	1MB	256KB

MultiTech

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mts_dragonfly	MultiTech Dragonfly	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F411RE 0MHz	512KB	128KB	
mts_mdot_f405rg	MultiTech mDot	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F411RE 0MHz	512KB	128KB	
mts_mdot_f411re	MultiTech mDot F411	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32F411RE 0MHz	512KB	128KB	
xdot_1151cc	MultiTech xDot	ST STM32	<i>Black Magic Probe, J-LINK, ST-LINK</i>	STM32L151C 0MHz	256KB	32KB	

NGX Technologies

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
blueboard_lpc	1NGX4 Technologies BlueBoard-LPC11U24	NXP LPC	<i>Black Magic Probe, J-LINK</i>	LPC11U24	48MHz	32KB	8KB

NXP

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lpc11c24	NXP LPC11C24	NXP LPC	<i>Black Magic Probe, J-LINK</i>	LPC11C24	48MHz	32KB	8KB
lpc11u24	NXP mbed LPC11U24	NXP LPC	<i>CMSIS-DAP (on-board), Black Magic Probe, J-LINK</i>	LPC11U24	48MHz	32KB	8KB
lpc11u24	ARM mbed LPC11U24 (+CAN)	NXP LPC	<i>CMSIS-DAP (on-board), Black Magic Probe, J-LINK</i>	LPC11U24	48MHz	32KB	8KB
lpc11u34	NXP LPC11U34	NXP LPC	<i>Black Magic Probe, J-LINK</i>	LPC11U34	48MHz	40KB	8KB
lpc11u37	NXP LPC11U37	NXP LPC	<i>Black Magic Probe, J-LINK</i>	LPC11U37	48MHz	128KB	10KB
lpc11u68	LPCXpresso11U68	NXP LPC	<i>CMSIS-DAP (on-board), Black Magic Probe, J-LINK</i>	LPC11U68	50MHz	256KB	36KB
lpc1549	NXP LPCXpresso1549	NXP LPC	<i>Black Magic Probe, J-LINK</i>	LPC1549	72MHz	256KB	36KB
lpc1768	NXP mbed LPC1768	NXP LPC	<i>CMSIS-DAP (on-board), Black Magic Probe, J-LINK</i>	LPC1768	96MHz	512KB	64KB
lpc54114	NXP LPCXpresso54114	NXP LPC	<i>CMSIS-DAP (on-board), J-LINK</i>	LPC54114J256 10MHz	256KB	192KB	
lpc546xx	NXP LPCXpresso54608	NXP LPC	<i>J-LINK (on-board)</i>	LPC54608ET51 80MHz	512KB	200KB	
lpc812	NXP LPC800-MAX	NXP LPC	<i>CMSIS-DAP (on-board), Black Magic Probe, J-LINK</i>	LPC812	30MHz	16KB	4KB
lpc824	LPCXpresso824-MAX	NXP LPC	<i>CMSIS-DAP (on-board), Black Magic Probe, J-LINK</i>	LPC824	30MHz	32KB	8KB

NodeMCU

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
nodemcu32S	NodeMCU 32S	Espressif 32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

Nordic

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
nrf51_dk	Nordic nRF51-DK	Nordic nRF51	CMSIS-DAP (on-board), J-LINK (on-board), Black Magic Probe, ST-LINK	NRF51	22MHz	256KB	32KB
nrf51_dongle	Nordic nRF51-Dongle	Nordic nRF51	CMSIS-DAP (on-board), J-LINK (on-board)	NRF51	22MHz	256KB	32KB
nrf51_mkit	iNordic nRF51822-mKIT	Nordic nRF51	CMSIS-DAP (on-board)	NRF51	26MHz	128KB	16KB
nrf52840_dk	Nordic nRF52840-DK	Nordic nRF52	CMSIS-DAP (on-board), J-LINK (on-board), Black Magic Probe, ST-LINK	NRF52840	40MHz	1MB	256KB
nrf52_dk	Nordic nRF52-DK	Nordic nRF52	CMSIS-DAP (on-board), J-LINK (on-board), Black Magic Probe, ST-LINK	NRF52832	40MHz	512KB	64KB

OLIMEX

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp32-e	OLIMEX ESP32-EVB	Espressif 32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB
esp32-g	OLIMEX ESP32-GATEWAY	Espressif 32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

OSHChip

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
oshchip	OS-HChip	Nordic nRF51	Black Magic Probe, J-LINK, ST-LINK	NRF51822	32MHz	256KB	32KB

RFduino

ID	Name	Platform	Debug	MCU	Fre-quency	Flash	RAM
rfduino	RF-duino	Nordic nRF51	<i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF51822	16MHz	128KB	8KB

RedBearLab

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
redBearLab	RedBearLab nRF51822	Nordic nRF51	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF51822	16MHz	256KB	16KB
redBearLabBLE	RedBearLab BLE Nano 1.5	Nordic nRF51	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF51822	16MHz	256KB	32KB
redbear_ble	RedBearLab BLE Nano 2	Nordic nRF52	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF52832	32MHz	512KB	64KB
redbear_ble	RedBearLab Blend 2	Nordic nRF52	<i>CMSIS-DAP</i> (on-board), <i>Black Magic Probe, J-LINK, ST-LINK</i>	NRF52832	32MHz	512KB	64KB

RoboticsBrno

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
alkses	Alks ESP32	Espres-sif 32	<i>Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY</i>	ESP32	240MHz	4MB	320KB

RushUp

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
cloud_jam	RushUp Cloud-JAM	ST STM32	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32F401RE	16MHz	512KB	96KB
cloud_jam	RushUp Cloud-JAM L4	ST STM32	<i>ST-LINK</i> (default, on-board), <i>Black Magic Probe, J-LINK</i>	STM32L476RBT6	16MHz	1MB	128KB

SODAQ

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
sodaq_automono	SODAQ Autonomo	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD21J18A	48MHz	256KB	32KB
sodaq_explorer	SODAQ Explorer	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD21J18A	48MHz	256KB	32KB
sodaq_one	SODAQ ONE	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD21G18A	48MHz	256KB	32KB

ST

ID	Name	Platform	Debug
disco_f030r8	ST STM32F0308DISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f051r8	ST STM32F0DISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f100rb	ST STM32VLDISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f303vc	ST STM32F3DISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f334c8	ST 32F3348DISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f401vc	ST 32F401CDISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f407vg	ST STM32F4DISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f411ve	ST 32F411EDISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f413zh	ST 32F413HDISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f429zi	ST 32F429IDISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f469ni	ST 32F469IDISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f746ng	ST 32F746GDISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_f769ni	ST 32F769IDISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_l053c8	ST 32L0538DISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_l072cz_lrwan1	ST DISCO-L072CZ-LRWAN1	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_l1152rb	ST STM32LLDISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_l475vg_iot01a	ST DISCO-L475VG-IOT01A	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
disco_l476vg	ST 32L476GDISCOVERY	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
eval_l073z	ST STM32L073Z-EVAL	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f030r8	ST Nucleo F030R8	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f031k6	ST Nucleo F031K6	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f042k6	ST Nucleo F042K6	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f070rb	ST Nucleo F070RB	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f072rb	ST Nucleo F072RB	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f091rc	ST Nucleo F091RC	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f103rb	ST Nucleo F103RB	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f207zg	ST Nucleo F207ZG	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f302r8	ST Nucleo F302R8	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f303k8	ST Nucleo F303K8	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f303re	ST Nucleo F303RE	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f303ze	ST Nucleo F303ZE	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f334r8	ST Nucleo F334R8	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f401re	ST Nucleo F401RE	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f410rb	ST Nucleo F410RB	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f411re	ST Nucleo F411RE	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK

Table 13 – continued from previous page

ID	Name	Platform	Debug
nucleo_f412zg	ST Nucleo F412ZG	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f413zh	ST Nucleo F413ZH	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f429zi	ST Nucleo F429ZI	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f446re	ST Nucleo F446RE	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f446ze	ST Nucleo F446ZE	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f746zg	ST Nucleo F746ZG	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_f767zi	ST Nucleo F767ZI	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_l011k4	ST Nucleo L011K4	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_l031k6	ST Nucleo L031K6	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_l053r8	ST Nucleo L053R8	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_l073rz	ST Nucleo L073RZ	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_l152re	ST Nucleo L152RE	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_l432kc	ST Nucleo L432KC	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_l476rg	ST Nucleo L476RG	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK
nucleo_l496zg	ST Nucleo L496ZG	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK

SainSmart

ID	Name	Platform	Debug	MCU	Fre- quency	Flash	RAM
sainSmartDue	SainSmart Due (Programming Port)	Atmel SAM	Black Magic Probe, J-LINK	AT91SAM3X8EMHz	512KB	32KB	
sainSmartDue	SainSmart Due (USB Native Port)	Atmel SAM	Black Magic Probe, J-LINK	AT91SAM3X8EMHz	512KB	32KB	

Samsung

ID	Name	Platform	Debug	MCU	Fre- quency	Flash	RAM
artik_053	Samsung ARTIK053	Samsung ARTIK	FTDI Chip (default)	S5JT200	320MHz	8MB	1.25MB

SeeedStudio

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
seeedArchBLE	Seeed Arch BLE	Nordic nRF51	CMSIS-DAP (on-board), Black Magic Probe, J-LINK, ST-LINK	NRF51822	16MHz	128KB	16KB
seeedArchLink	Seeed Arch Link	Nordic nRF51	CMSIS-DAP (on-board), Black Magic Probe, J-LINK, ST-LINK	NRF51822	16MHz	256KB	16KB
seeedArchMax	Seeed Arch Max	ST STM32	ST-LINK (default, on-board), Black Magic Probe, J-LINK	STM32F407VET6	168MHz	512KB	192KB
seeedArchPro	Seeed Arch Pro	NXP LPC	CMSIS-DAP (on-board)	LPC1768	96MHz	512KB	64KB
seeedTinyBLE	Seeed Tiny BLE	Nordic nRF51	CMSIS-DAP (on-board), Black Magic Probe, J-LINK, ST-LINK	NRF51822	16MHz	256KB	16KB

Semtech

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
mote_1152rc	NAMote72	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32L152RCT6	32MHz	256KB	32KB

SiFive

ID	Name	Plat-form	Debug	MCU	Fre-quency	Flash	RAM
freedom-e300-hifive1	Hi-Five1	RISC-V	FTDI Chip (on-board)	FE310	320MHz	16MB	16KB

Silicon Labs

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
efm32gg_stk3700	Silicon Labs EFM32GG-STK3700 (Giant Gecko)	Silicon Labs EFM32	J-LINK (on-board), Black Magic Probe	EFM32GG9018MHz	1MB	128KB	
efm32hg_stk3400	Silicon Labs SLSTK3400A USB-enabled (Happy Gecko)	Silicon Labs EFM32	J-LINK (on-board), Black Magic Probe	EFM32HG3224MHz	64KB	8KB	
efm32lg_stk3600	Silicon Labs EFM32LG-STK3600 (Leopard Gecko)	Silicon Labs EFM32	J-LINK (on-board), Black Magic Probe	EFM32LG9018MHz	256KB	32KB	
efm32pg_stk3401	Silicon Labs SLSTK3401A (Pearl Gecko)	Silicon Labs EFM32	J-LINK (on-board), Black Magic Probe	EFM32PG1B200MHz	256KB	32KB	
efm32wg_stk3800	Silicon Labs EFM32WG-STK3800 (Wonder Gecko)	Silicon Labs EFM32	J-LINK (on-board), Black Magic Probe	EFM32WG9018MHz	256KB	32KB	
efm32zg_stk3200	Silicon Labs EFM32ZG-STK3200 (Zero Gecko)	Silicon Labs EFM32	J-LINK (on-board), Black Magic Probe	EFM32ZG2224MHz	32KB	4KB	

Solder Splash Labs

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
dipcortexm	Solder Splash Labs Dip-Cortex M0	NXP LPC	Black Magic Probe, J-LINK	LPC11U24	450MHz	32KB	8KB
lpc1347	DipCortex M3	NXP LPC	J-LINK	LPC1347	72MHz	64KB	12KB

SparkFun

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
sparkfun_samd21	SparkFun SAMD21 Breakout Dev	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD21G18MHz	256KB	32KB	
sparkfun_samd21	SparkFun SAMD21 Mini Breakout	Atmel SAM	Atmel-ICE, Black Magic Probe, J-LINK	SAMD21G18MHz	256KB	32KB	

SparkFun Electronics

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
esp32t	SparkFun ESP32 Thing	Espressif 32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

Switch Science

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
hrm1017	Switch Science mbed HRM1017	Nordic nRF51	CMSIS-DAP (on-board)	NRF51822	16MHz	256KB	16KB
lpc1114fn28	Switch Science mbed LPC1114FN28	NXP LPC	CMSIS-DAP (on-board), Black Magic Probe, J-LINK	LPC1114FN28	16MHz	32KB	4KB
ssci824	Switch Science mbed LPC824	NXP LPC	CMSIS-DAP (on-board), Black Magic Probe, J-LINK	LPC824	30MHz	32KB	8KB
ty51822r3	Switch Science mbed TY51822r3	Nordic nRF51	CMSIS-DAP (on-board)	NRF51822	32MHz	256KB	32KB

TI

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lplm4f120h5	TI LaunchPad (Stellaris) w/ lm4f120 (80MHz)	TI TIVA	TI-ICDI (on-board)	LPLM4F120H5	80MHz	256KB	32KB
1pmmsp430f5	TI LaunchPad MSP-EXP430F5529LP	TI MSP430	MSP Debug (on-board)	MSP430F5529	16MHz	128KB	8KB
1pmmsp430fr4	TI LaunchPad MSP-EXP430FR4133LP	TI MSP430	MSP Debug (on-board)	MSP430FR4133	38MHz	15KB	2KB
1pmmsp430fr5	TI FraunchPad MSP-EXP430FR5739LP	TI MSP430	MSP Debug (on-board)	MSP430FR5739	16MHz	16KB	512B
1pmmsp430fr5	TI LaunchPad MSP-EXP430FR5969LP	TI MSP430	MSP Debug (on-board)	MSP430FR5969	98MHz	64KB	2KB
1pmmsp430fr6	TI LaunchPad MSP-EXP430FR6989LP	TI MSP430	MSP Debug (on-board)	MSP430FR6989	98MHz	127KB	2KB
1pmmsp430g25	TI LaunchPad MSP-EXP430G2553LP	TI MSP430	MSP Debug (on-board)	MSP430G2553	16MHz	16KB	512B
1ptm4c1230c3	TI LaunchPad (Tiva C) w/ tm4c123 (80MHz)	TI TIVA	TI-ICDI (on-board)	LPTM4C1230C	80MHz	256KB	32KB
1ptm4c1294n	TI LaunchPad (Tiva C) w/ tm4c129 (120MHz)	TI TIVA	TI-ICDI (on-board)	LPTM4C1294N	120MHz	1MB	256KB

TTGO

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
ttgo-lora32v1	TTGO v1 LoRa32-OLED V1	Espressif 32	- Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

Taida Century

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
stct_nrf52_miniboard	Taida v Century nRF52 mini board	Nordic nRF52	Black Magic Probe, J-LINK, ST-LINK	NRF52830	4MHz	512KB	64KB

Teensy

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
teensy31	Teensy 3.1 / 3.2	Teensy	J-LINK	MK20DX256	72MHz	256KB	64KB
teensy35	Teensy 3.5	Teensy	J-LINK	MK64FX512	120MHz	512KB	192KB
teensy36	Teensy 3.6	Teensy	J-LINK	MK66FX1M0	180MHz	1MB	256KB
teensylc	Teensy LC	Teensy	J-LINK	MKL26Z64	48MHz	62KB	8KB

ThaiEasyElec

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
espino32	Espresif Pino32	Espresso 32	- Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

VNG

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
vbluno51	VNG VBLUNO51	Nordic nRF51	CMSIS-DAP (on-board)	NRF51822	16MHz	128KB	32KB

WEMOS

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lolin32	WEMOS LOLIN32	Espresso-Sif32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	240MHz	4MB	320KB
lolin_d32	WEMOS LOLIN D32	Espresso-Sif32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	240MHz	4MB	320KB
lolin_d32_pro	WEMOS LOLIN D32 PRO	Espresso-Sif32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	240MHz	4MB	320KB
wemosbat32	WeMos WiFi & Bluetooth Battery	Espresso-Sif32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	240MHz	4MB	320KB

WIZNet

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
wizwiki_w7500	WIZwiki-W7500	WIZNet W7500	CMSIS-DAP (on-board), J-LINK	WIZ-NET7500	48MHz	128KB	48KB
wizwiki_w7500_eco	WIZwiki-W7500ECO	WIZNet W7500	CMSIS-DAP (on-board), J-LINK	WIZ-NET7500ECO	48MHz	128KB	48KB
wizwiki_w7500p	WIZwiki-W7500P	WIZNet W7500	CMSIS-DAP (on-board), J-LINK	WIZ-NET7500P	48MHz	128KB	48KB

Waveshare

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
waveshare_ble400	Waveshare BLE400	Nordic nRF51	Black Magic Probe, J-LINK, ST-LINK	NRF51822	232MHz	256KB	32KB

Xilinx

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
coreplexip-e31	Factory E310 Arty (Artix-7) FPGA Dev Kit	RISC-V	Olimex ARM-USB-TINY-H	E31	320MHz	16MB	256MB
coreplexip-e51	Factory Arty (Artix-7) FPGA Dev Kit	RISC-V	Olimex ARM-USB-TINY-H	E51	1500MHz	16MB	256MB

XinaBox

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
xinabox_Xin02 aBox CW02	Xin02 Espresso 32	ESP32	Mini-Module FT2232H, Olimex ARM-USB-OCD-H, Olimex ARM-USB-OCD, Olimex ARM-USB-TINY-H, Olimex ARM-USB-TINY	ESP32	2240MHz	4MB	320KB

ng-beacon

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
ng_beacon_ng-beacon	ng-beacon	Nordic nRF51	Black Magic Probe, J-LINK, ST-LINK	NRF51822	32MHz	256KB	32KB

u-blox

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
mbed_connect_Mbed in Connect Cloud	Mbed in Connect Cloud	ST STM32	CMSIS-DAP (on-board), Black Magic Probe, J-LINK, ST-LINK	STM32F439I	18MHz	2MB	256KB
mtb_ublox_ubloxw2_ODIN-W2	ubloxw2 ODIN-W2	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F439I	18MHz	2MB	256KB
ublox_c030_u_blox_C030-N211_IoT_Starter_Kit	u-blox C030-N211 IoT Starter Kit	ST STM32	Black Magic Probe, CMSIS-DAP, J-LINK, ST-LINK	STM32F437N	80MHz	1MB	256KB
ublox_c030_u_blox_C030-U201_IoT_Starter_Kit	u-blox C030-U201 IoT Starter Kit	ST STM32	Black Magic Probe, CMSIS-DAP, J-LINK, ST-LINK	STM32F437N	80MHz	1MB	256KB
ublox_evk_nihoxb1_EVK-NINA-B1	nihoxb1 EVK-NINA-B1	Nordic nRF52	J-LINK (on-board), Black Magic Probe, ST-LINK	NRF52832	64MHz	512KB	64KB
ublox_evk_ubloxw2_EVK-ODIN-W2	ubloxw2 EVK-ODIN-W2	ST STM32	Black Magic Probe, J-LINK, ST-LINK	STM32F439I	18MHz	2MB	256KB
ubloxc027_u-blox_C027	u-blox C027	NXP LPC	CMSIS-DAP (on-board), Black Magic Probe, J-LINK	LPC1768	96MHz	512KB	64KB

y5 design

ID	Name	Platform	Debug	MCU	Frequency	Flash	RAM
lpc11u35_y5_mbug	LPC11U35 mbug	NXP LPC	Black Magic Probe, J-LINK	LPC11U3	38MHz	64KB	10KB
nrf51822_y5_mbug	nRF51822 mbug	Nordic nRF51	CMSIS-DAP (on-board), Black Magic Probe, J-LINK, ST-LINK	NRF51822	26MHz	256KB	16KB

1.17 PIO Unit Testing

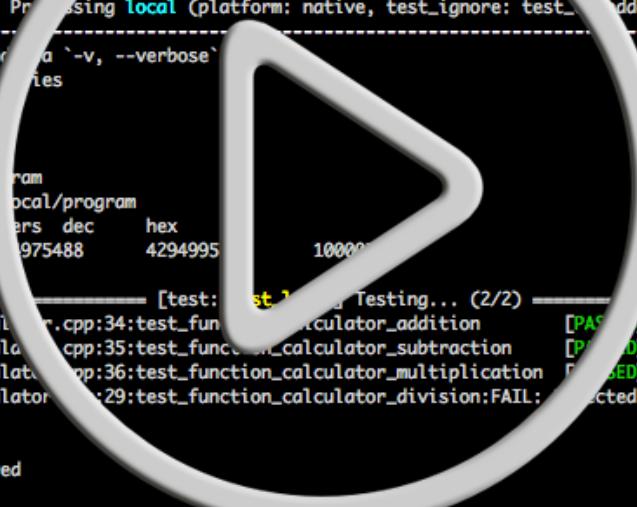
New in version 3.0: (PIO Plus)

Unit Testing ([wiki](#)) is a software testing method by which individual units of source code, sets of one or more MCU program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Unit testing finds problems early in the development cycle.

Contents

- [*Demo*](#)
- [*Tutorials and Examples*](#)
 - [*Tutorials*](#)
 - [*Project Examples*](#)
- [*Configuration*](#)
- [*Test Types*](#)
 - [*Desktop*](#)
 - [*Embedded*](#)
- [*Test Runner*](#)
 - [*Local*](#)
 - [*Remote*](#)
- [*Workflow*](#)
 - [*Shared Code*](#)
- [*API*](#)
- [*User Guide \(CLI\)*](#)

1.17.1 Demo



```

----- [test::test_embedded] Testing... (3/3) -----
If you don't see any output for the first 10 secs, please reset board (press reset button)

test/test_embedded/test_calculator.cpp:36:test_function_calculator_addition      [PASSED]
test/test_embedded/test_calculator.cpp:37:test_function_calculator_subtraction    [PASSED]
test/test_embedded/test_calculator.cpp:38:test_function_calculator_multiplication [PASSED]
test/test_embedded/test_calculator.cpp:31:test_function_calculator_division:FAIL: Expected 32 Was 32
-----
4 Tests 1 Failures 0 Ignored

----- [test::test_local] Building... (1/2) =
[Wed Sep 7 15:36:06 2016] Processing local (platform: native, test_ignore: test_embedded, piofile: .pioenvs/local/program)
Verbose mode can be enabled via `--v, --verbose`
Collected 1 compatible libraries
Looking for dependencies.
Library Dependency Graph
|-- <calculator>
Linking .pioenvs/local/program
Calculating size .pioenvs/local/program
__TEXT __DATA __OBJC __OBJC0 __DATA __OBJC0 __DATA __OBJC0
16384 4096 0 4975488 4294995 10000
-----
----- [test::test_local] Testing... (2/2) =
test/test_local/test_calculator.cpp:34:test_function_calculator_addition      [PASSED]
test/test_local/test_calculator.cpp:35:test_function_calculator_subtraction    [PASSED]
test/test_local/test_calculator.cpp:36:test_function_calculator_multiplication [PASSED]
test/test_local/test_calculator.cpp:29:test_function_calculator_division:FAIL: Expected 32 Was 32
-----
4 Tests 1 Failures 0 Ignored
FAIL

----- [TEST SUMMARY] =
test:test_common/env:uno          [PASSED]
test:test_common/env:local         [PASSED]
test:test_embedded/env:uno         [FAILED]
test:test_embedded/env:local       [IGNORED]
test:test_local/env:uno           [IGNORED]
test:test_local/env:local         [FAILED]

----- [FAILED] Took 20.40 seconds -----

```

Demo of Local & Embedded: Calculator.

1.17.2 Tutorials and Examples

Tutorials

- [Unit Testing of a “Blink” Project](#)
- [STM32Cube HAL and Nucleo-F401RE: debugging and unit testing](#)
- [ThingForward: Start Embedded Testing with PlatformIO](#)
- [ThingForward: Embedded Testing with PlatformIO - Part 2](#)
- [ThingForward: Embedded Testing with PlatformIO – Part 3: Remoting](#)
- [ThingForward: Embedded Testing with PlatformIO – Part 4: Continuous Integration](#)
- [ThingForward, Webinar: Unit Testing for Embedded with PlatformIO and Qt Creator](#)

Project Examples

- Embedded: Wiring Blink
- Local & Embedded: Calculator
- Labmet Weather Station
- PlatformIO Remote Unit Testing Example

For the other examples and source code please follow to [PlatformIO Unit Testing Examples](#) repository.

1.17.3 Configuration

PIO Unit Testing Engine can be configured from *Project Configuration File platformio.ini*

- *test_filter*
- *test_ignore*
- *test_port*
- *test_speed*
- *test_transport*

1.17.4 Test Types

Desktop

PIO Unit Testing Engine builds a test program for a host machine using *Native* development platform. This test could be run only with the desktop or *Continuous Integration* VM instance.

Note: PlatformIO does not install any toolchains automatically for *Native* and requires GCC toolchain to be installed on your host machine. Please open Terminal and check that the `gcc` command is installed.

Embedded

PIO Unit Testing Engine builds a special firmware for a target device (board) and program it. Then, it connects to this device using configured Serial *test_port* and communicate via *test_transport*. Finally, it runs test on an embedded side, collects results, analyzes them and provides a summary on a host machine side (desktop).

Note: Please note that the **PIO Unit Testing Engine** uses the first available Serial/UART implementation (depending on a *framework*) as a communication interface between the **PIO Unit Testing Engine** and target device. If you use `Serial` in your project libraries, please wrap/hide Serial-based blocks with `#ifndef UNIT_TEST` macro.

Also, you can create custom *test_transport* and implement base interface.

1.17.5 Test Runner

Test Runner allows you to process specific environments or ignore a test using “Glob patterns”. You can also ignore a test for specific environments using a *test_ignore* option from *Project Configuration File platformio.ini*.

Local

Allows you to run a test on a host machine or on a target device (board) which is directly connected to the host machine. In this case, you need to use the [platformio test](#) command.

Remote

Allows you to run test on a remote machine or remote target device (board) without having to depend on OS software, extra software, SSH, VPN or opening network ports. Remote Unit Testing works in pair with [PIO Remote](#). In this case, you need to use the special command [platformio remote test](#).

PlatformIO supports multiple [Continuous Integration](#) systems where you can run unit tests at the each integration stage. See real [PlatformIO Remote Unit Testing Example](#).

1.17.6 Workflow

1. Create PlatformIO project using the [platformio init](#) command. For Desktop Unit Testing (on a host machine), you need to use [Native](#).

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

;
; Embedded platforms
;

[env:uno]
platform = atmelavr
framework = arduino
board = uno

[env:nodemcu]
platform = espressif8266
framework = arduino
board = nodemcuv2

;
; Desktop platforms (Win, Mac, Linux, Raspberry Pi, etc)
; See https://platformio.org/platforms/native
;

[env:native]
platform = native
```

2. Create a `test` folder in a root of your project. See [test_dir](#).
3. Write a test using [API](#). Each test is a small independent program/firmware with its own `main()` or `setup()` / `loop()` functions. Test should start with `UNITY_BEGIN()` and finish with `UNITY_END()` calls.

Warning: If your board does not support software reset via Serial.DTR/RTS, you should add >2 seconds delay before UNITY_BEGIN(). That time is needed to establish a ``Serial connection between a host machine and a target device.

```
delay(2000); // for Arduino framework
wait(2); // for ARM mbed framework
UNITY_BEGIN();
```

4. Place a test in the test directory. If you have more than one test, split them into sub-folders. For example, test/test_1/*.[c, cpp, h], test_N/*.[c, cpp, h], etc. If there is no such directory in the test``folder, then |PIOUTE| will treat the source code of ``test folder as SINGLE test.
5. Run test using the *platformio test* command.

Shared Code

PIO Unit Testing Engine does not build source code from *src_dir* folder by default. If you have a shared/common code between your “main” and “test” programs, you have 2 options:

1. **RECOMMENDED.** We recommend to split a program source code into multiple components and place them into *lib_dir* (project’s private libraries and components). *Library Dependency Finder (LDF)* will find these libraries automatically and include to build process. You will need to include any library/component header file in your test or program source code via #include <MyComponent.h>.

See [Local & Embedded: Calculator](#) example. We have “calculator” component in *lib_dir* folder and include it in tests and main program using #include <calculator.h>.

2. Manually instruct PlatformIO to build source code from *src_dir* folder using *test_build_project_src* option in *Project Configuration File platformio.ini*:

```
[env:myenv]
platform = ...
test_build_project_src = true
```

This is very useful if you unit test independent library where you can’t split source code.

Warning: Please note that you will need to use #ifdef UNIT_TEST and #endif guard to hide non-test related source code. For example, own main() or setup() / loop() functions.

1.17.7 API

Summary of the Unity Test API:

- Running Tests
 - RUN_TEST(func, linenum)
- Ignoring Tests
 - TEST_IGNORE()
 - TEST_IGNORE_MESSAGE (message)
- Aborting Tests

- TEST_PROTECT()
- TEST_ABORT()
- Basic Validity Tests
 - TEST_ASSERT_TRUE(condition)
 - TEST_ASSERT_FALSE(condition)
 - TEST_ASSERT(condition)
 - TEST_ASSERT_UNLESS(condition)
 - TEST_FAIL()
 - TEST_FAIL_MESSAGE(message)
- Numerical Assertions: Integers
 - TEST_ASSERT_EQUAL_INT(expected, actual)
 - TEST_ASSERT_EQUAL_INT8(expected, actual)
 - TEST_ASSERT_EQUAL_INT16(expected, actual)
 - TEST_ASSERT_EQUAL_INT32(expected, actual)
 - TEST_ASSERT_EQUAL_INT64(expected, actual)
 - TEST_ASSERT_EQUAL_UINT(expected, actual)
 - TEST_ASSERT_EQUAL_UINT8(expected, actual)
 - TEST_ASSERT_EQUAL_UINT16(expected, actual)
 - TEST_ASSERT_EQUAL_UINT32(expected, actual)
 - TEST_ASSERT_EQUAL_UINT64(expected, actual)
 - TEST_ASSERT_EQUAL_HEX(expected, actual)
 - TEST_ASSERT_EQUAL_HEX8(expected, actual)
 - TEST_ASSERT_EQUAL_HEX16(expected, actual)
 - TEST_ASSERT_EQUAL_HEX32(expected, actual)
 - TEST_ASSERT_EQUAL_HEX64(expected, actual)
 - TEST_ASSERT_EQUAL_HEX8_ARRAY(expected, actual, elements)
 - TEST_ASSERT_EQUAL(expected, actual)
 - TEST_ASSERT_INT_WITHIN(delta, expected, actual)
- Numerical Assertions: Bitwise
 - TEST_ASSERT_BITS(mask, expected, actual)
 - TEST_ASSERT_BITS_HIGH(mask, actual)
 - TEST_ASSERT_BITS_LOW(mask, actual)
 - TEST_ASSERT_BIT_HIGH(mask, actual)
 - TEST_ASSERT_BIT_LOW(mask, actual)
- Numerical Assertions: Floats
 - TEST_ASSERT_FLOAT_WITHIN(delta, expected, actual)

- TEST_ASSERT_EQUAL_FLOAT(expected, actual)
- TEST_ASSERT_EQUAL_DOUBLE(expected, actual)
- String Assertions
 - TEST_ASSERT_EQUAL_STRING(expected, actual)
 - TEST_ASSERT_EQUAL_STRING_LEN(expected, actual, len)
 - TEST_ASSERT_EQUAL_STRING_MESSAGE(expected, actual, message)
 - TEST_ASSERT_EQUAL_STRING_LEN_MESSAGE(expected, actual, len, message)
- Pointer Assertions
 - TEST_ASSERT_NULL(pointer)
 - TEST_ASSERT_NOT_NULL(pointer)
- Memory Assertions
 - TEST_ASSERT_EQUAL_MEMORY(expected, actual, len)

1.17.8 User Guide (CLI)

1.18 Cloud & Desktop IDE

1.18.1 PlatformIO IDE

PlatformIO IDE is the next-generation integrated development environment for IoT.

We provide official open-source packages (plugins, extensions) for the most popular IDEs and text editors.

PlatformIO IDE for VSCode

PlatformIO IDE is the next-generation integrated development environment for IoT.

- Cross-platform build system without external dependencies to the OS software:
 - 400+ embedded boards
 - 20+ development platforms
 - 10+ frameworks
- *PIO Unified Debugger*
- *PIO Remote*
- *PIO Unit Testing*
- C/C++ Intelligent Code Completion
- C/C++ Smart Code Linter for rapid professional development
- Library Manager for the hundreds popular libraries
- Multi-projects workflow with multiple panes
- Themes support with dark and light colors
- Serial Port Monitor

- Built-in Terminal with *PlatformIO Core* and CLI tool (`pio`, `platformio`)

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Python, PHP, Go) and runtimes (such as .NET and Unity)

The screenshot shows the PlatformIO Debugger integrated into Visual Studio Code. The left sidebar contains toolbars for DEBUG, PL, and MEMORY. The main area displays the `WiFi.cpp` file with code related to WiFi provisioning. The right side shows the memory dump and assembly dump panes. The bottom features a terminal window executing a task.

```

DEBUG ▶ PL ⚙️ 🖨 WiFi.cpp ✎ Memory [0x42000800+1] ... WiFiClass::startProvision.dbgasm ⏪ ⏴ ⏵ ⏹ ⏸ 🔍
VARIABLES Local
  strM2MAPConfig: {...}
  this@entry: 0x200007c0 <WiFi>
  ssid@entry: 0x20007fa8 "wifi1"
  ssid@entry: 0x20007fa8 ...
  url@entry: 0xd67c <fini+752>
  url@entry: 0xd67c <fini...
  channel: 1 '\001'
  channel@entry: 1 '\001'
Global
Static
WATCH mac: [6]
strM2MAPConfig.au8DHCPServerIP[0]: 192 '\300'
strM2MAPConfig.au8DHCPServerIP[1]: 168 '\250'
strM2MAPConfig.au8DHCPServerIP[2]: 1 '\001'
CALL STACK PAUSED ON ST...
WiFiClass::startProvision@...
WiFiClass::beginProvision@...
WiFiClass::beginProvision@...
setup@0x000002d6 setup...
main@0x0006b82 main.d...
BREAKPOINTS WiFi.cpp .piolib... [588]
PERIPHERALS PORT [0x41004400]
RTC [0x40001400]
SERCOM0 [0x4200080...]
I2CM [0x0]
I2CS [0x0]
SPI [0x0]
REGISTERS pc = 0x00000726
xPSR = 0x21000000
  Negative Flag (N) = 0
  Zero Flag (Z) = 0
  Carry or borrow flag (...)
  Overflow Flag (V) = 0
MEMORY
DISASSEMBLY
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
> Executing task: platformio device monitor <
--- Miniterm on /dev/cu.usbmodemFA142 9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+H followed by Ctrl+H ---
Configuring WiFi shield/module...
Starting in provisioning mode...

```

Contents

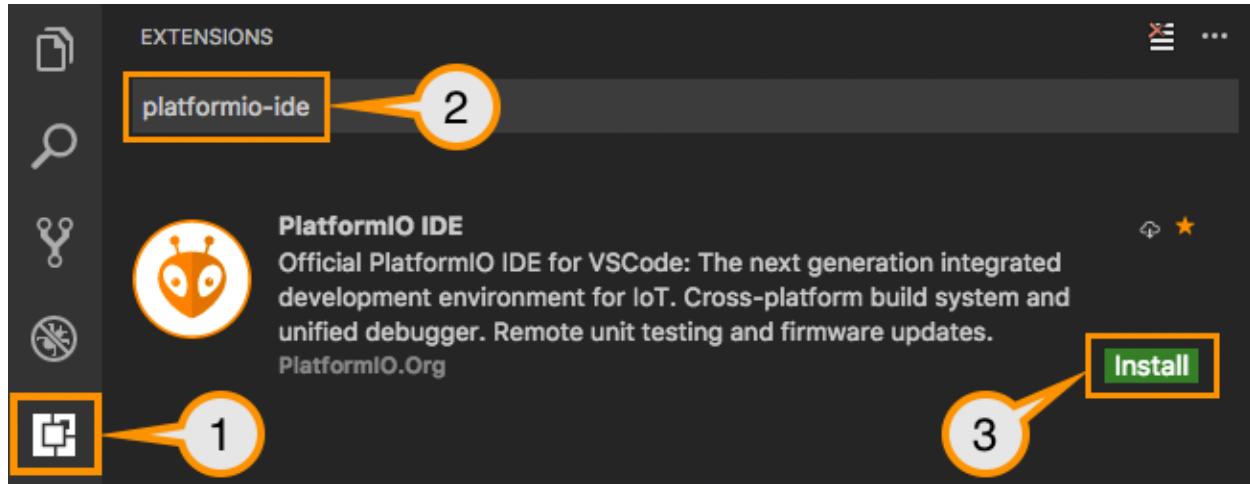
- Installation*
- Quick Start*
 - Setting Up the Project*
- PlatformIO Toolbar*

- *Key Bindings*
- *Task Runner*
- *Custom Tasks*
- *Serial Port Monitor*
- *Install Shell Commands*
- *Settings*
 - `platformio-ide.useBuiltinPIOCore`
 - `platformio-ide.useDevelopmentPIOCore`
 - `platformio-ide.autoRebuildAutocompleteIndex`
 - `platformio-ide.forceUploadAndMonitor`
 - `platformio-ide.customPATH`
 - `platformio-ide.updateTerminalPathConfiguration`
 - `platformio-ide.activateOnlyOnPlatformIOProject`
 - `platformio-ide.defaultToolbarBuildAction`
 - `platformio-ide.autoCloseSerialMonitor`
- *Known issues*
 - `PackageManager` is unable to install tool
- *Changelog*

Installation

Note: Please note that you do not need to install *PlatformIO Core* separately if you are going to use *PlatformIO IDE for VSCode*. *PlatformIO Core* is built into PlatformIO IDE and you will be able to use it within PlatformIO IDE Terminal.

0. Download and install official Microsoft Visual Studio Code. PlatformIO IDE is built on top of it
1. Open VSCode Package Manager
2. Search for official `platformio-ide` extension
3. Install PlatformIO IDE.

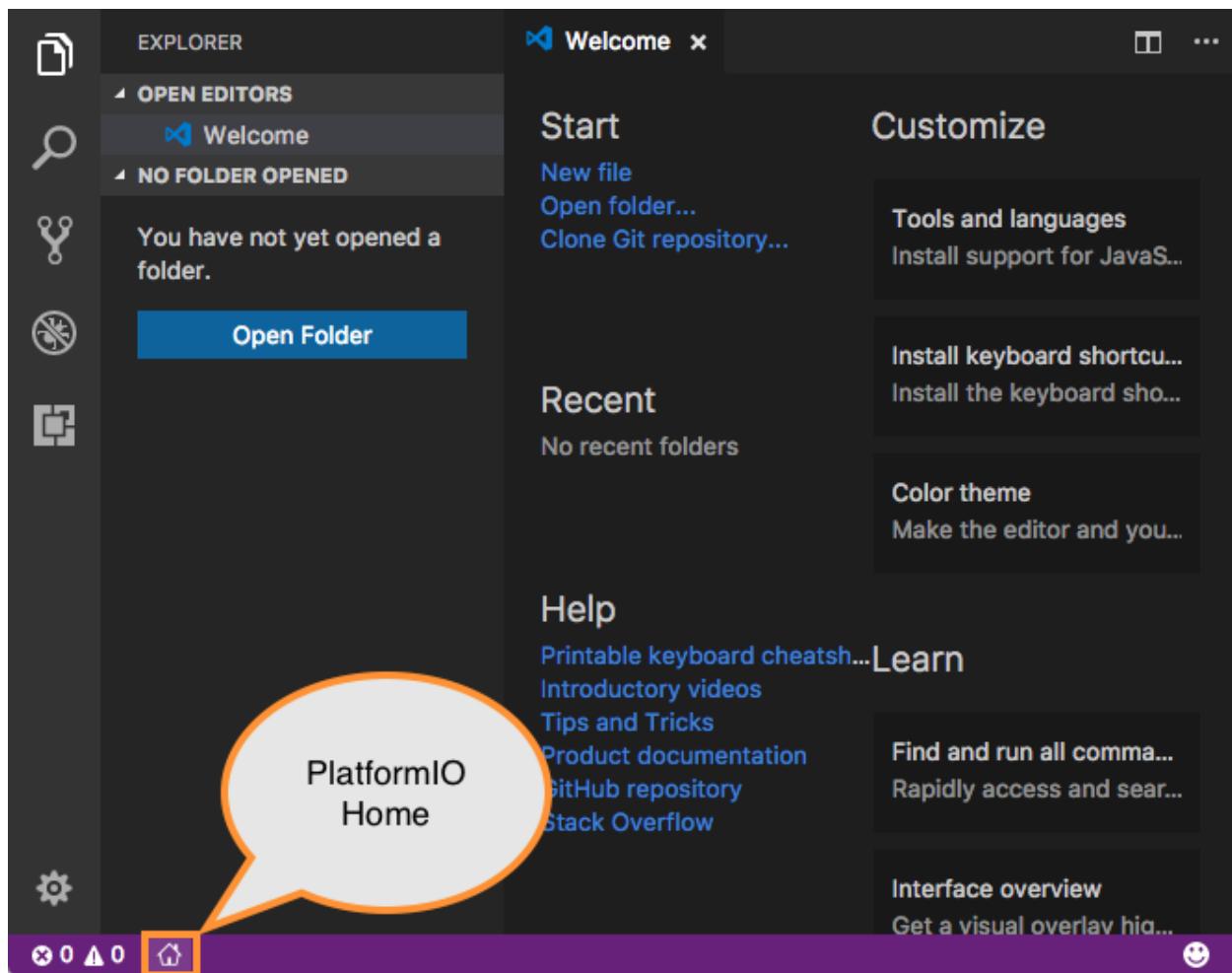


Quick Start

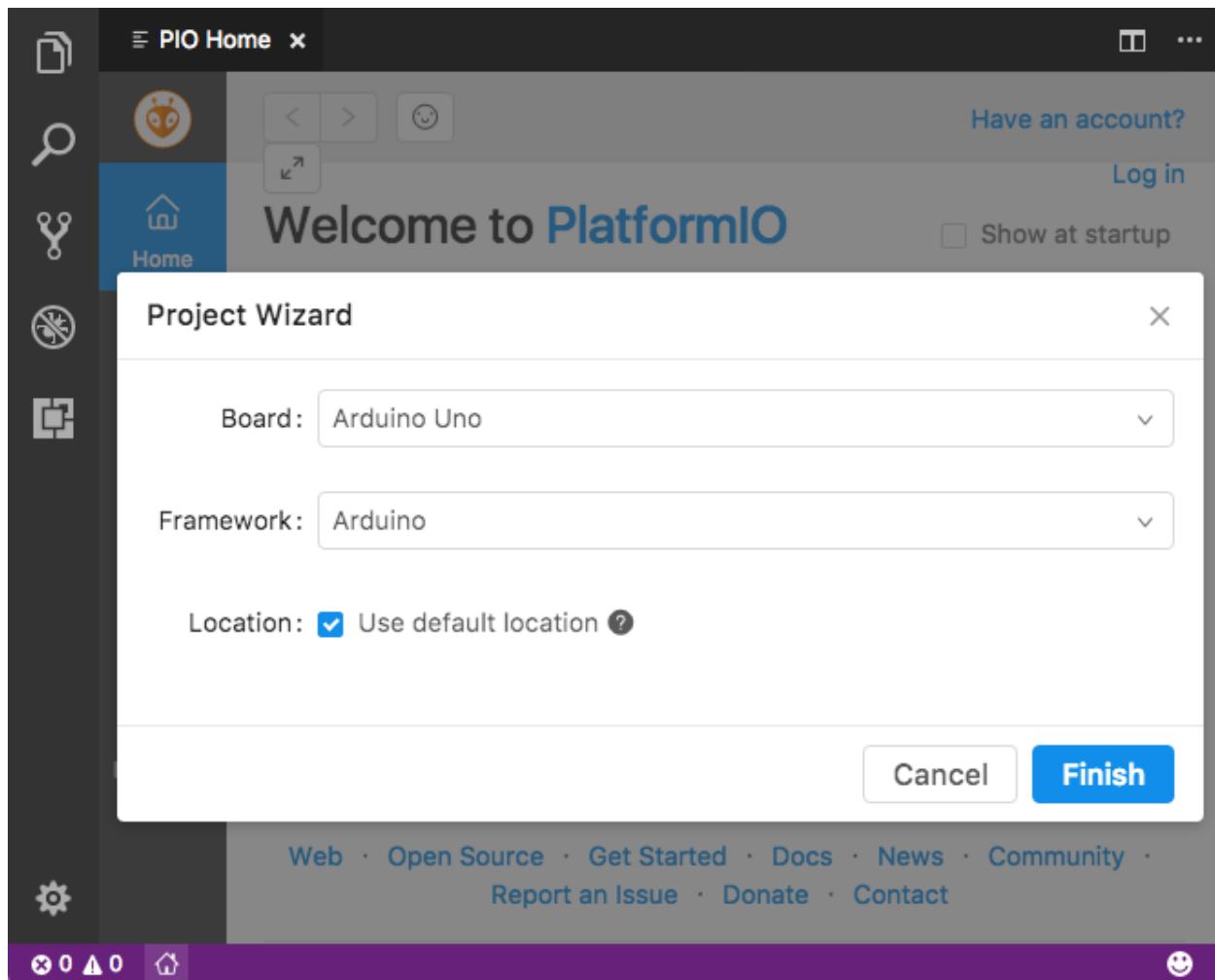
This tutorial introduces you to the basics of PlatformIO IDE workflow and shows you a creation process of a simple “Blink” example. After finishing you will have a general understanding of how to work with projects in the IDE.

Setting Up the Project

1. Click on “PlatformIO Home” button on the bottom *PlatformIO Toolbar*



2. Click on “New Project”, select a board and create new PlatformIO Project



3. Open `main.cpp` file from `src` folder and replace its contents with the next:

Warning: The code below works only in pair with Arduino-based boards. Please follow to [PlatformIO Project Examples](#) repository for other pre-configured projects.

```
/** 
 * Blink
 *
 * Turns on an LED on for one second,
 * then off for one second, repeatedly.
 */
#include "Arduino.h"

// Set LED_BUILTIN if it is not defined by Arduino framework
// #define LED_BUILTIN 13

void setup()
{
    // initialize LED digital pin as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}
```

(continues on next page)

(continued from previous page)

```

void loop()
{
    // turn the LED on (HIGH is the voltage level)
    digitalWrite(LED_BUILTIN, HIGH);

    // wait for a second
    delay(1000);

    // turn the LED off by making the voltage LOW
    digitalWrite(LED_BUILTIN, LOW);

    // wait for a second
    delay(1000);
}

```

```

EXPLORER          main.cpp x
OPEN EDITORS
  main.cpp src
  170901-141209-UNO
    .pioenvs
    .vscode
    lib
    src
      main.cpp
platformio.ini

4  * Turns on an LED on for one second,
5  * then off for one second, repeatedly.
6  */
7  #include "Arduino.h"
8
9  // Set LED_BUILTIN if it is not defined by Arduino
10 // #define LED_BUILTIN 13
11
12 void setup()
13 {
14     // initialize LED digital pin as an output.
15     pinMode(LED_BUILTIN, OUTPUT);
16 }
17
18 void loop()
19 {
20     // turn the LED on (HIGH is the voltage level)
21     digitalWrite(LED_BUILTIN, HIGH);

22     // wait for a second
23     delay(1000);

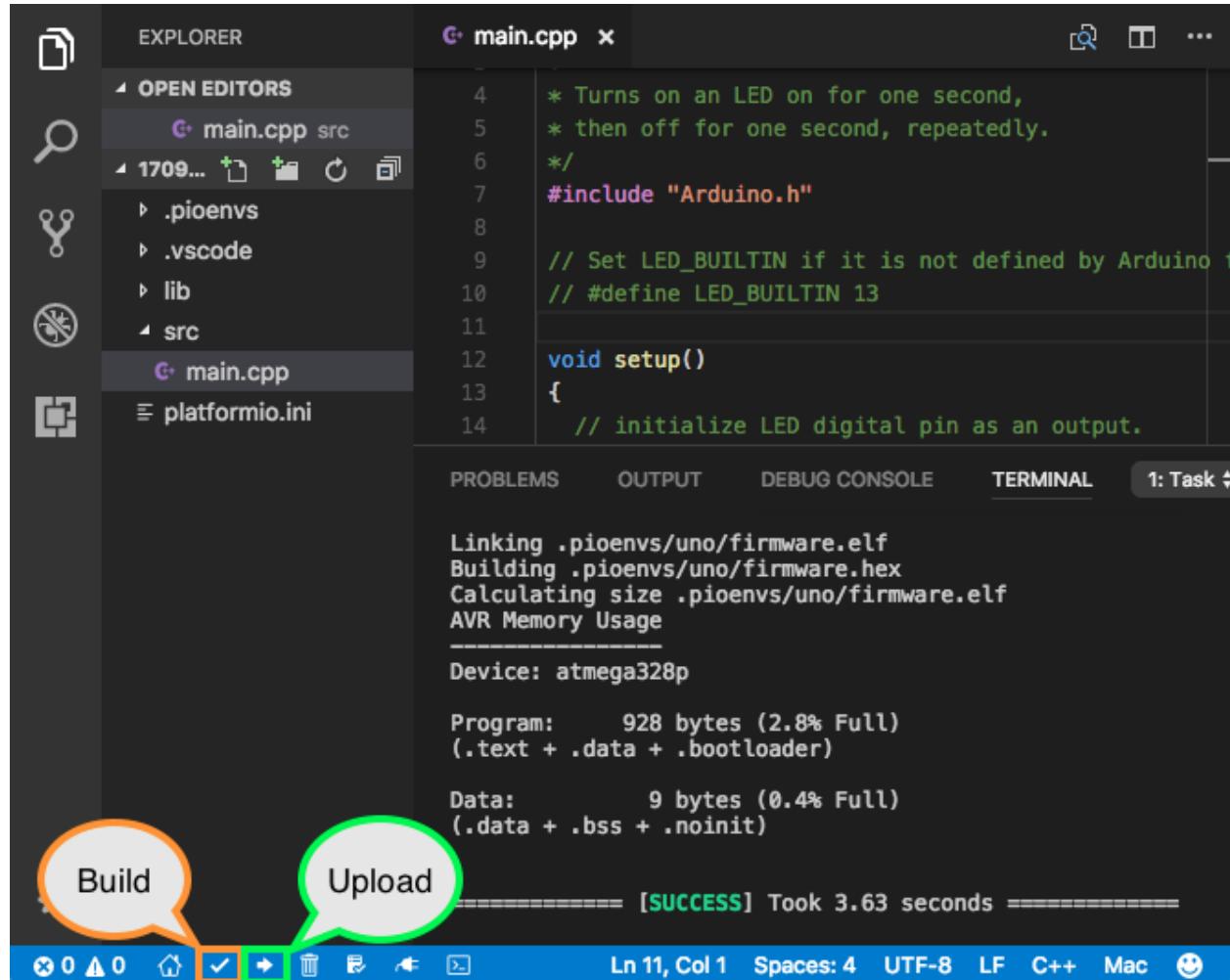
24     // turn the LED off by making the voltage LOW
25     digitalWrite(LED_BUILTIN, LOW);

26     // wait for a second
27     delay(1000);
28
29

```

Ln 12, Col 9 Spaces: 4 UTF-8 LF C++ Mac ☺

4. Build your project with `ctrl+alt+b` hotkey (see all Key Bindings in “User Guide” section below) or using “Build” button on the *PlatformIO Toolbar*



Learn more about *PlatformIO Toolbar* and other commands (Upload, Clean, Serial Monitor) below.

Happy coding with PlatformIO!

PlatformIO Toolbar

PlatformIO IDE Toolbar is located in VSCode Status Bar (left corner) and contains quick access buttons for the popular commands. Each button contains hint (delay mouse on it).



1. [PlatformIO Home](#)
2. PlatformIO: Build
3. PlatformIO: Upload
4. [PIO Remote](#)
5. PlatformIO: Clean
6. [PIO Unit Testing](#)

7. Run a task... (See “Task Runner” below)
8. *Serial Port Monitor*
9. PIO Terminal

Key Bindings

- **ctrl+alt+b / cmd-shift-b / ctrl-shift-b** Build Project
- **cmd-shift-d / ctrl-shift-d** Debug project
- **ctrl+alt+u** Upload Firmware
- **ctrl+alt+s** Open *Serial Port Monitor*

Task Runner

PlatformIO IDE provides base tasks `Menu > Tasks` (Build, Upload, Clean, Monitor, etc) and custom tasks per *Project Configuration File platformio.ini* environment (`[env:***]`). A default behavior is to use Terminal Panel for presentation. Also, we use dedicated panel per unique task.

PlatformIO IDE provides own Problems Matcher named `$platformio`. You can use it later if decide to change base task settings.

You can override existing task with own presentation options. For example, let configure PlatformIO Task Runner to use NEW Terminal panel per each “Build” command:

1. Please click on “gear” icon near “Build” task in `Menu > Tasks`
2. Replace template in `tasks.json` with this code

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "type": "PlatformIO",
      "task": "Monitor",
      "problemMatcher": [
        "$platformio"
      ],
      "presentation": {
        "panel": "new"
      }
    }
  ]
}
```

See more options in [official VSCode documentation](#).

Custom Tasks

Custom tasks can be added to `tasks.json` file located in `.vscode` folder in the root of project. Please read official documentation [Tasks in VSCode](#).

This simple example demonstrates a custom build process in verbose mode. There are a lot of other commands, please read more about [PlatformIO Core](#) and its commands ([User Guide](#)).

```
{  
    "version": "2.0.0",  
    "tasks": [  
        {  
            "type": "shell",  
            "command": "platformio",  
            "args": [  
                "run",  
                "--verbose"  
            ],  
            "problemMatcher": [  
                "$platformio"  
            ],  
            "label": "PlatformIO: Verbose Build"  
        }  
    ]  
}
```

Serial Port Monitor

You can customize Serial Port Monitor using *Monitor options* in *Project Configuration File platformio.ini*:

- *monitor_port*
- *monitor_speed*
- *monitor_rts*
- *monitor_dtr*

Example:

```
[env:esp32dev]  
platform = espressif32  
framework = arduino  
board = esp32dev  
  
; Custom Serial Monitor port  
monitor_port = /dev/ttyUSB1  
  
; Custom Serial Monitor speed (baud rate)  
monitor_speed = 115200
```

Install Shell Commands

Please navigate to PIO Core *Install Shell Commands*.

Settings

How to configure VSCode settings?

```
platformio-ide.useBuiltInPIOCore
```

Use built-in *PlatformIO Core*, default value is true.

platformio-ide.useDevelopmentPIOCore

Use development version of *PlatformIO Core*, default value is `false`.

platformio-ide.autoRebuildAutocompleteIndex

Automatically rebuild C/C++ Project Index when *Project Configuration File platformio.ini* is changed or when new libraries are installed, default value is `true`.

platformio-ide.forceUploadAndMonitor

Force “Upload and Monitor” task for Upload (`platformio-ide.upload`) command, default value is `false`.

platformio-ide.customPATH

Custom PATH for `platformio` command. Paste here the result of `echo $PATH` (Unix) / `echo %PATH%` (Windows) command by typing into your system terminal if you prefer to use custom version of *PlatformIO Core*, default value is `null`.

platformio-ide.updateTerminalPathConfiguration

Update Terminal configuration with patched PATH environment, default value is `true`.

platformio-ide.activateOnlyOnPlatformIOProject

Activate extension only when PlatformIO-based project (with *Project Configuration File platformio.ini*) is opened in workspace, default value is `false`.

platformio-ide.defaultToolbarBuildAction

Default action for “Build” button on *PlatformIO Toolbar*, default value is `release`. Possible values are `release` or `pre-debug`.

To eliminate a full project rebuilding before debugging, please change this value to `pre-debug`.

platformio-ide.autoCloseSerialMonitor

Automatically close *platformio device monitor* before uploading/testing, default value is `true`.

Known issues

PackageManager is unable to install tool

This is a known bug in VSCode Terminal [issue #61](#).

A temporary solution is to install packages using a system terminal (not VSCode Terminal). Please use “Solution 3: Run from Terminal” in FAQ > Package Manager > [\[Error 5\] Access is denied](#).

Now, back to VSCode.

Changelog

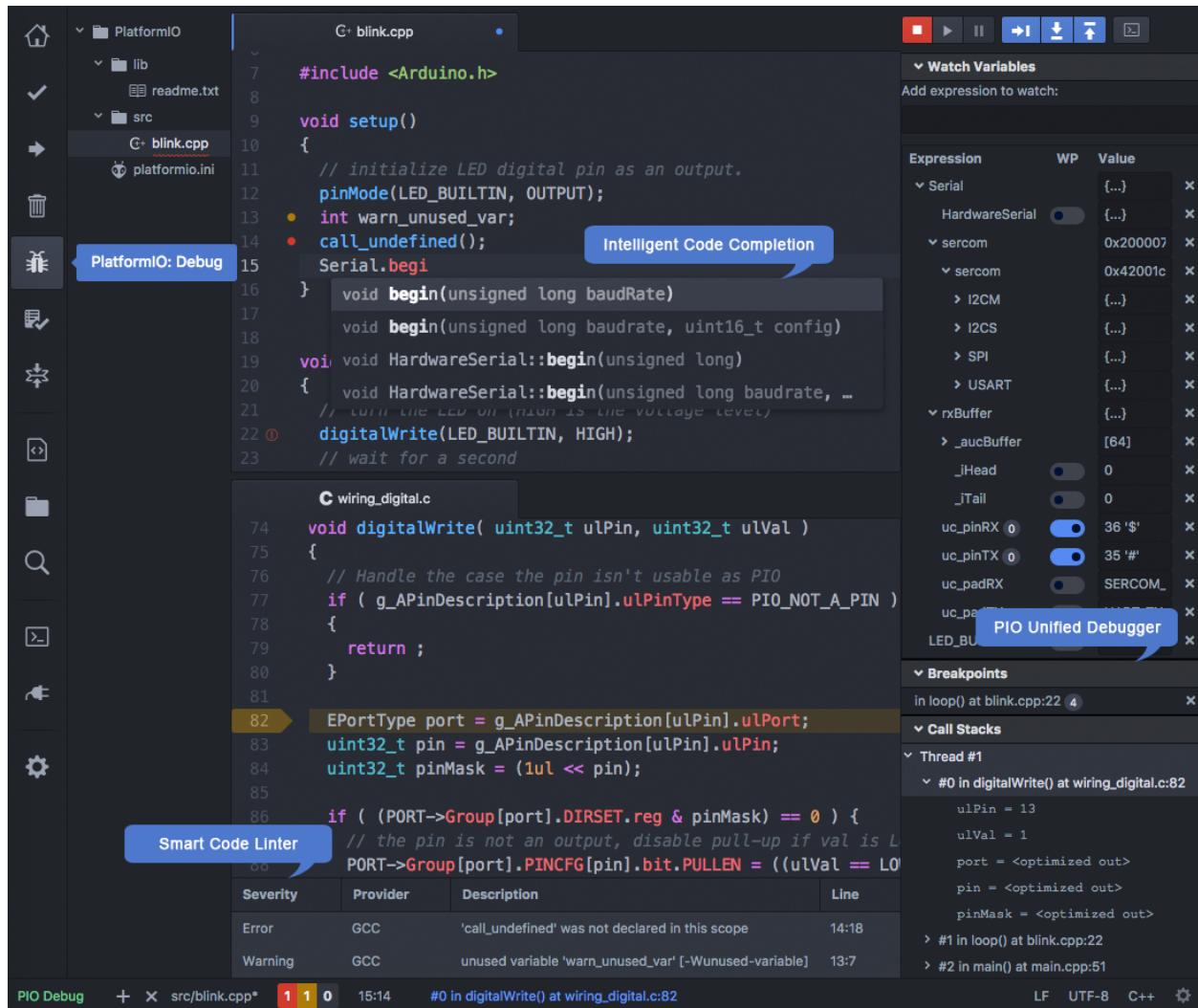
Please visit [releases](#) page.

PlatformIO IDE for Atom

PlatformIO IDE is the next-generation integrated development environment for IoT.

- Cross-platform build system without external dependencies to the OS software:
 - 400+ embedded boards
 - 20+ development platforms
 - 10+ frameworks
- [PIO Unified Debugger](#)
- [PIO Remote](#)
- [PIO Unit Testing](#)
- C/C++ Intelligent Code Completion
- C/C++ Smart Code Linter for rapid professional development
- Library Manager for the hundreds popular libraries
- Multi-projects workflow with multiple panes
- Themes support with dark and light colors
- Serial Port Monitor
- Built-in Terminal with [PlatformIO Core](#) and CLI tool (`pio`, `platformio`)

Atom is a text editor that's modern, approachable, yet hackable to the core—a tool you can customize to do anything but also use productively without ever touching a config file.



Contents

- *Installation*
 - *I. Atom*
 - *II. Clang for Intelligent Code Completion*
- *Quick Start*
 - *Launch*
 - *Setting Up the Project*
 - *Process Project*
- *Menu item PlatformIO*
- *PlatformIO Toolbar*
- *Building / Uploading / Targets*
- *Intelligent Code Completion*

- [Smart Code Linter](#)
- [Install Shell Commands](#)
- [Known issues](#)
 - [Smart Code Linter is disabled for Arduino files](#)
 - * [Convert Arduino file to C++ manually](#)
 - * [Force Arduino file as C++](#)
 - [Arch Linux: PlatformIO IDE Terminal issue](#)
- [Frequently Asked Questions](#)
 - [Keep build panel visible](#)
 - [Automatically save on build](#)
 - [Jump to Declaration](#)
 - [Code Formatting](#)
- [Uninstall Atom with PlatformIO IDE](#)
 - [Windows](#)
 - [macOS](#)
 - [Linux](#)
- [Articles / Manuals](#)
- [Changelog](#)

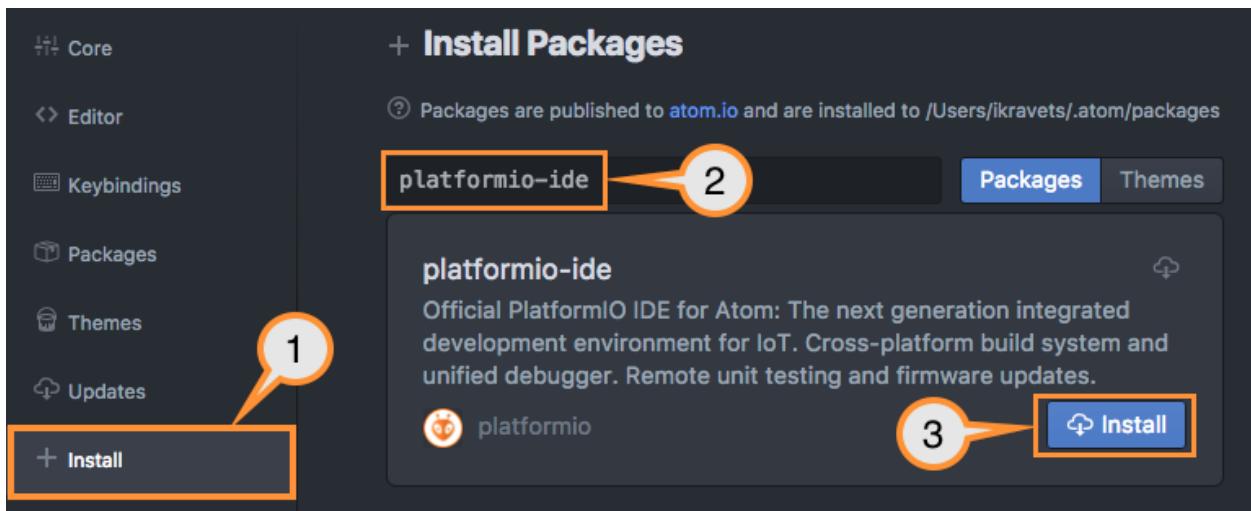
Installation

Note: Please note that you do not need to install [PlatformIO Core](#) separately if you are going to use [PlatformIO IDE for Atom](#). [PlatformIO Core](#) is built into PlatformIO IDE and you will be able to use it within PlatformIO IDE Terminal.

Also, PlatformIO IDE allows to install [PlatformIO Core](#) Shell Commands (pio, platformio) globally to your system via Menu: PlatformIO > Install Shell Commands.

I. Atom

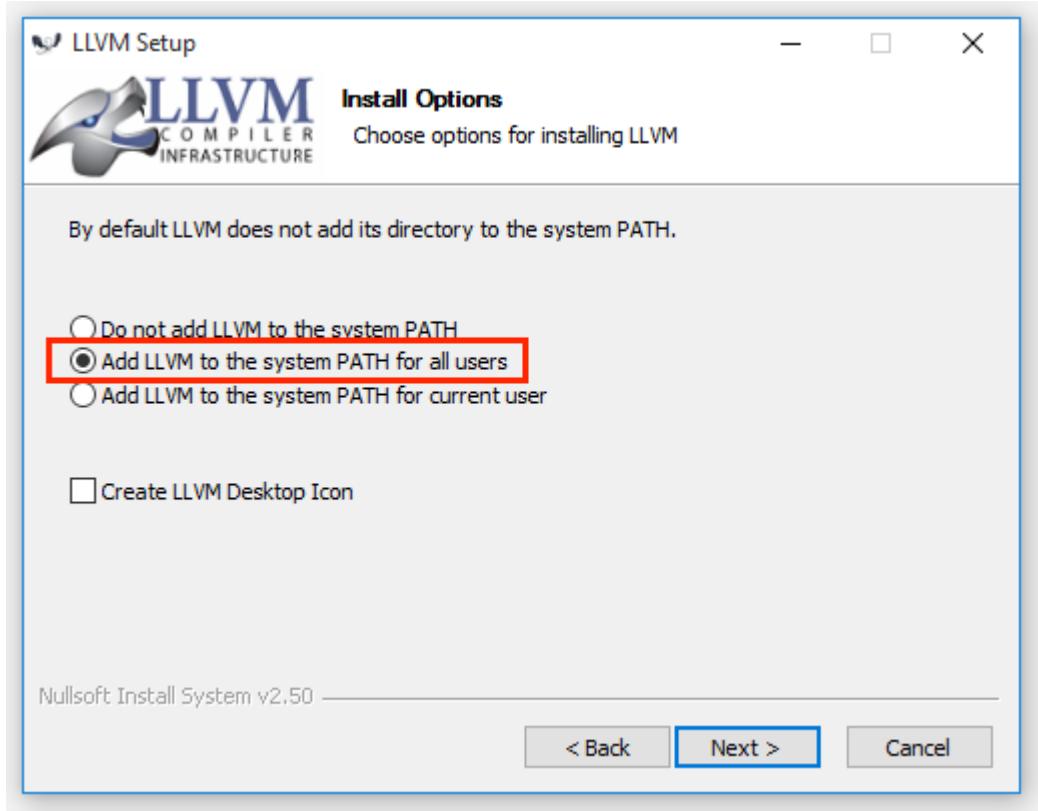
0. [Download](#) and install official GitHub's Atom text editor. PlatformIO IDE is built on top of it
1. **Open** Atom Package Manager
 - [Mac OS X](#), Menu: Atom > Preferences > Install
 - [Windows](#), Menu: File > Settings > Install
 - [Linux](#), Menu: Edit > Preferences > Install
2. **Search** for official platformio-ide package
3. **Install** PlatformIO IDE.



II. Clang for Intelligent Code Completion

PlatformIO IDE uses [Clang](#) for the Intelligent Code Completion. To check that clang is available in your system, please open Terminal and run `clang --version`. If clang is not installed, then **install it and restart Atom**:

- **Mac OS X:** [Install the latest Xcode](#) along with the latest Command Line Tools (they are installed automatically when you run clang in Terminal for the first time, or manually by running `xcode-select --install`)
- **Windows:** Download [Clang 3.9.1 for Windows](#). Please select “Add LLVM to the system PATH” option on the installation step.
 - [Clang 3.9.1 for Windows \(32-bit\)](#)
 - [Clang 3.9.1 for Windows \(64-bit\)](#)



Warning: PLEASE DO NOT INSTALL CLANG 4.0. TEMPORARY, WE SUPPORT ONLY CLANG 3.9.

If you see Failed to find MSBuild toolsets directory error in the installation console, please ignore it and press any key to close this window. PlatformIO IDE uses only Clang completion engine that should work after it without any problems.

- **Linux:** Using package managers: `apt-get install clang` or `yum install clang`.
- **Other Systems:** Download the latest [Clang](#) for the other systems.

Warning: If some libraries are not visible in [PlatformIO IDE for Atom](#) and Code Completion or Code Linting does not work properly, please perform Menu: PlatformIO > Rebuild C/C++ Project Index (Autocomplete, Linter)

Quick Start

This tutorial introduces you to the basics of PlatformIO IDE workflow and shows you a creation process of a simple “Blink” example. After finishing you will have a general understanding of how to work with projects in the IDE.

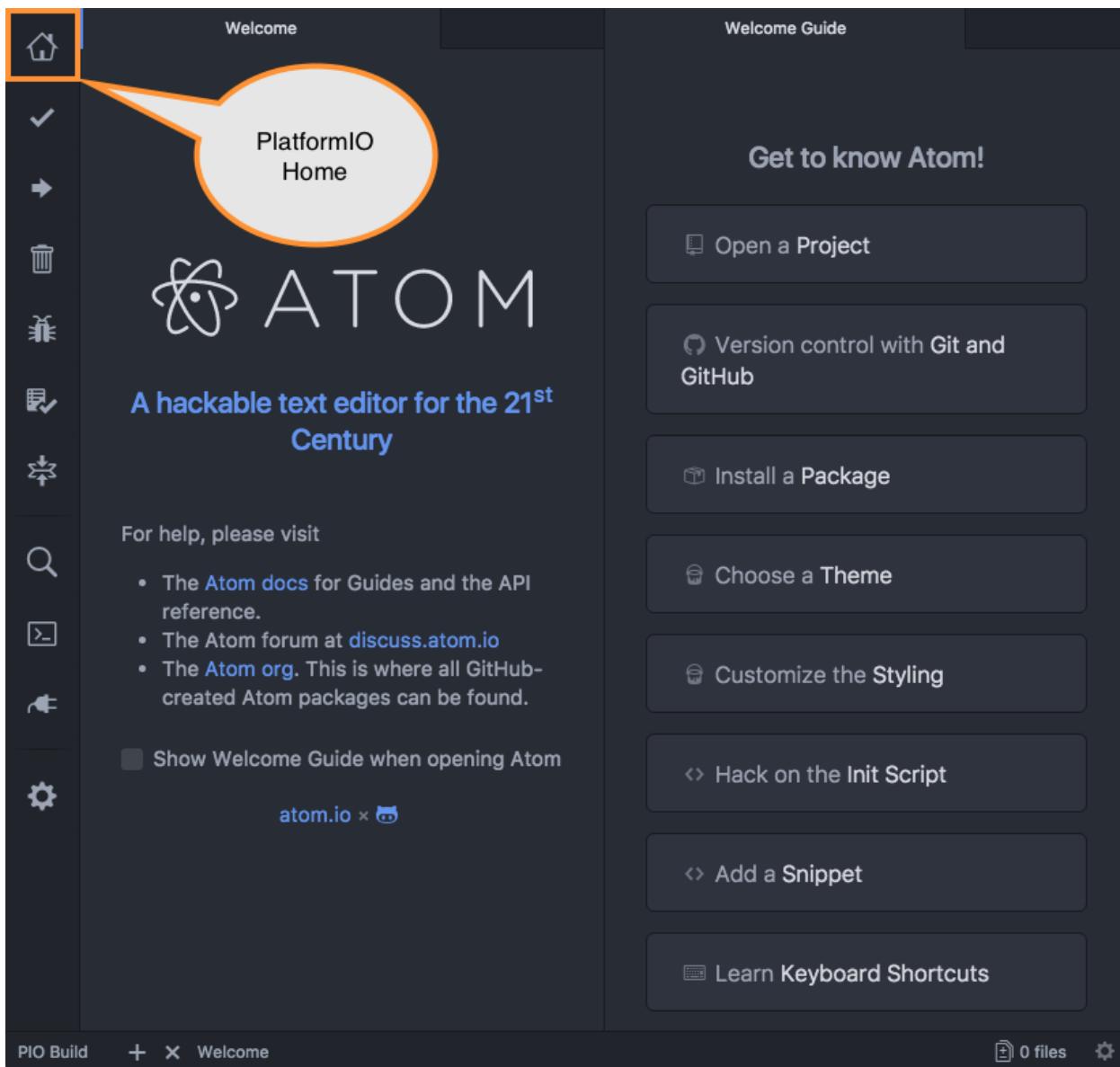
Launch

After installation, you launch PlatformIO IDE by opening Atom. Once Atom is opened, PlatformIO IDE auto installer will continue to install dependent packages and [PlatformIO Core](#). Please be patient and let the installation complete.

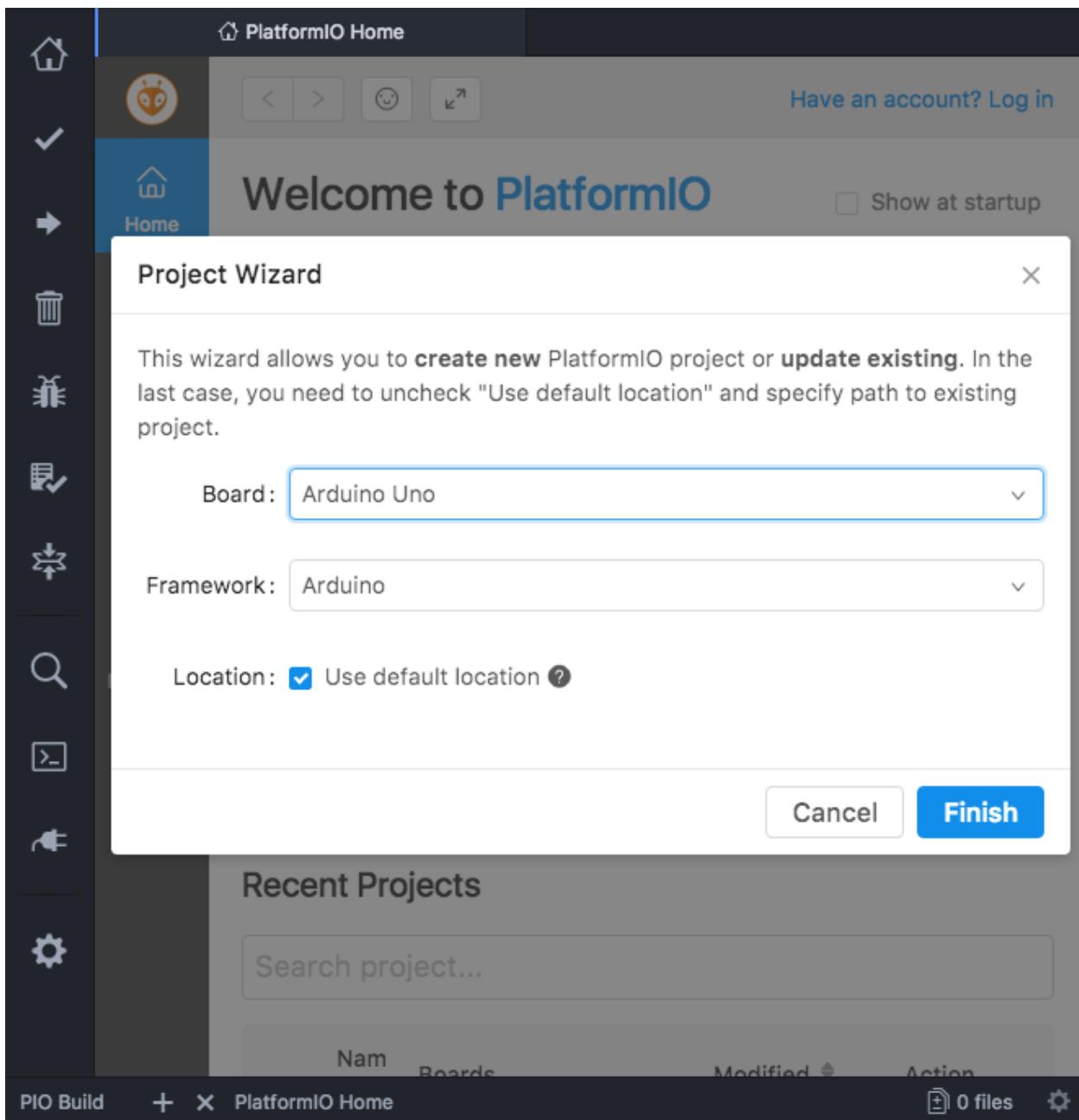
In the final result PlatformIO IDE will ask you to reload Atom window to apply installed components. Please click on Reload Now. After it PlatformIO IDE is ready for using. Happy coding!

Setting Up the Project

1. Click on “PlatformIO Home” button on the *PlatformIO Toolbar*



2. Click on “New Project”, select a board and create new PlatformIO Project



3. Open `main.cpp` file from `src` folder and replace its contents with the next:

Warning: The code below works only in pair with Arduino-based boards. Please follow to [PlatformIO Project Examples](#) repository for other pre-configured projects.

```
/** 
 * Blink
 *
 * Turns on an LED on for one second,
 * then off for one second, repeatedly.
 */
```

(continues on next page)

(continued from previous page)

```
#include "Arduino.h"

// Set LED_BUILTIN if it is not defined by Arduino framework
// #define LED_BUILTIN 13

void setup()
{
    // initialize LED digital pin as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
    // turn the LED on (HIGH is the voltage level)
    digitalWrite(LED_BUILTIN, HIGH);

    // wait for a second
    delay(1000);

    // turn the LED off by making the voltage LOW
    digitalWrite(LED_BUILTIN, LOW);

    // wait for a second
    delay(1000);
}
```

The screenshot shows the PlatformIO IDE interface. On the left is a toolbar with various icons: house, checkmark, right arrow, trash, bee, eye, download, magnifying glass, folder, plug, and gear. The main area has a title bar "Project" and a file tree under "170905-185203-un" containing "lib", "src" (with "main.cpp" and "platformio.ini"), and a "Project" icon. To the right is a code editor window titled "G+ main.cpp" with the following C++ code:

```

1  /**
2  * Blink
3  *
4  * Turns on an LED on for one second,
5  * then off for one second, repeatedly.
6  */
7 #include "Arduino.h"
8
9 // Set LED_BUILTIN if it is not defined by Arduino.h
10 // #define LED_BUILTIN 13
11
12 void setup()
13 {
14     // initialize LED digital pin as an output.
15     pinMode(LED_BUILTIN, OUTPUT);
16 }
17
18 void loop()
19 {
20     // turn the LED on (HIGH is the voltage level)
21     digitalWrite(LED_BUILTIN, HIGH);
22
23     // wait for a second
24     delay(1000);
25
26     // turn the LED off by making the voltage LOW
27     digitalWrite(LED_BUILTIN, LOW);
28 }

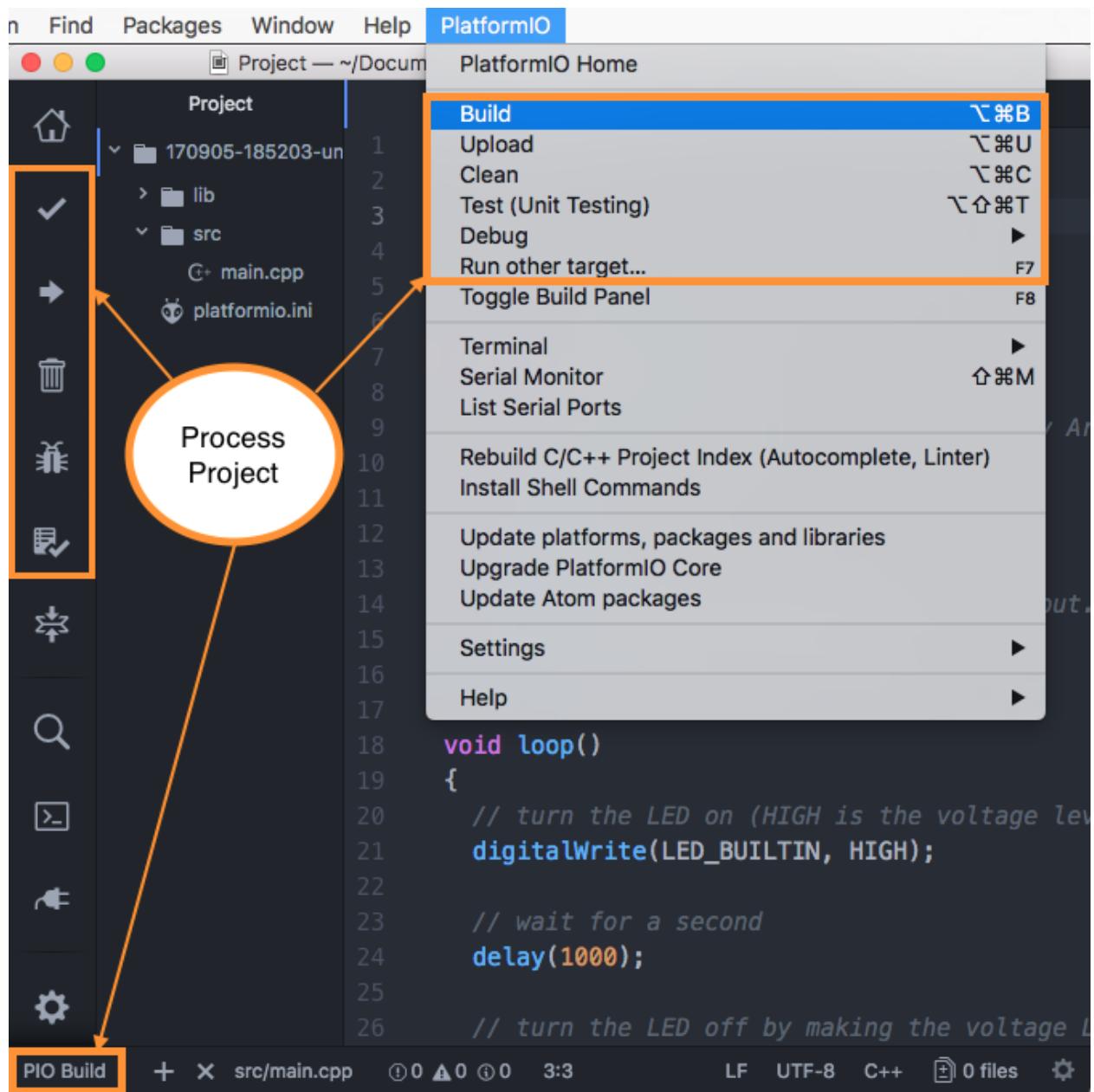
```

At the bottom, there are buttons for "PIO Build" (+), "src/main.cpp", file status (0 0 ▲ 0 0 0), and build status (3:3). On the right are buttons for LF, UTF-8, C++, 0 files, and settings.

Process Project

PlatformIO IDE proposes different ways to process project (build, clean, upload firmware, run other targets) using:

- *PlatformIO Toolbar*
- *Menu item PlatformIO*
- *Building / Uploading / Targets* and hotkeys



5. Run Build and you should see green “success” result in the building panel:

The screenshot shows the PlatformIO IDE interface. On the left is a toolbar with various icons: Home, Checkmark, Build (highlighted), Run, Delete, Find, Open, Save, and Settings. The central area is divided into sections: 'Project' (containing a folder named '170905-185203-un'), 'src' (containing 'main.cpp'), and 'platformio.ini'. A code editor window titled 'G+ main.cpp' displays the following C++ code:

```

1  /**
2  * Blink
3  *
4  * Turns on an LED on for one second,
5  * then off for one second, repeatedly.
6  */
7 #include "Arduino.h"
8
9 // Set LED_BUILTIN if it is not defined by Arduino
10 // #define LED_BUILTIN 13
11
12 void setup()
13 {
14
15 }

```

Below the code editor, the status bar shows the command 'platformio run' and a duration of '5.0 s'. To the right of the status bar are three buttons: a lightning bolt (Upload), a trash can (Delete), and a close (X). The main workspace displays 'AVR Memory Usage' for an 'atmega328p' device, showing memory usage details:

```

Device: atmega328p

Program: 928 bytes (2.8% Full)
(.text + .data + .bootloader)

Data: 9 bytes (0.4% Full)
(.data + .bss + .noinit)

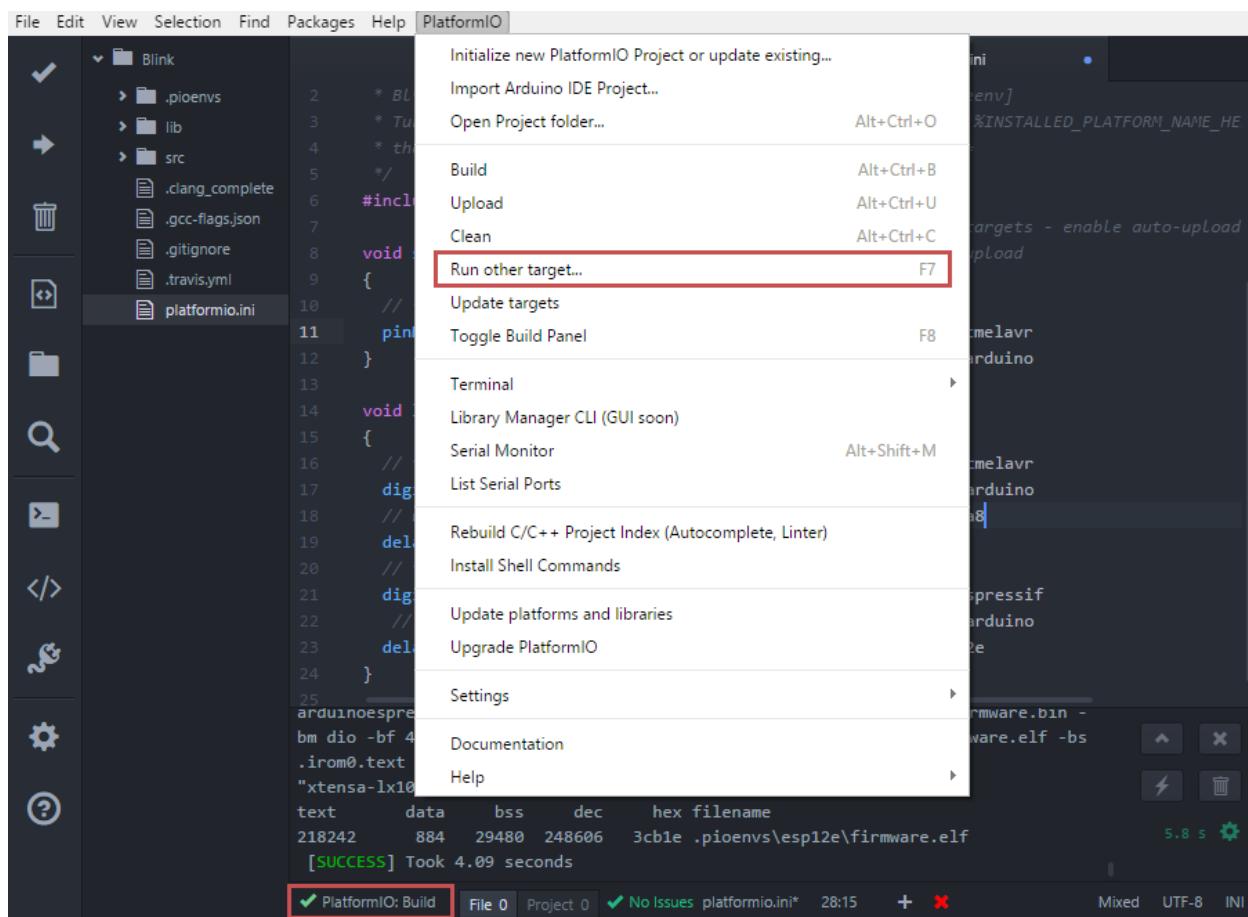
===== [SUCCESS] Took 4.01 seconds

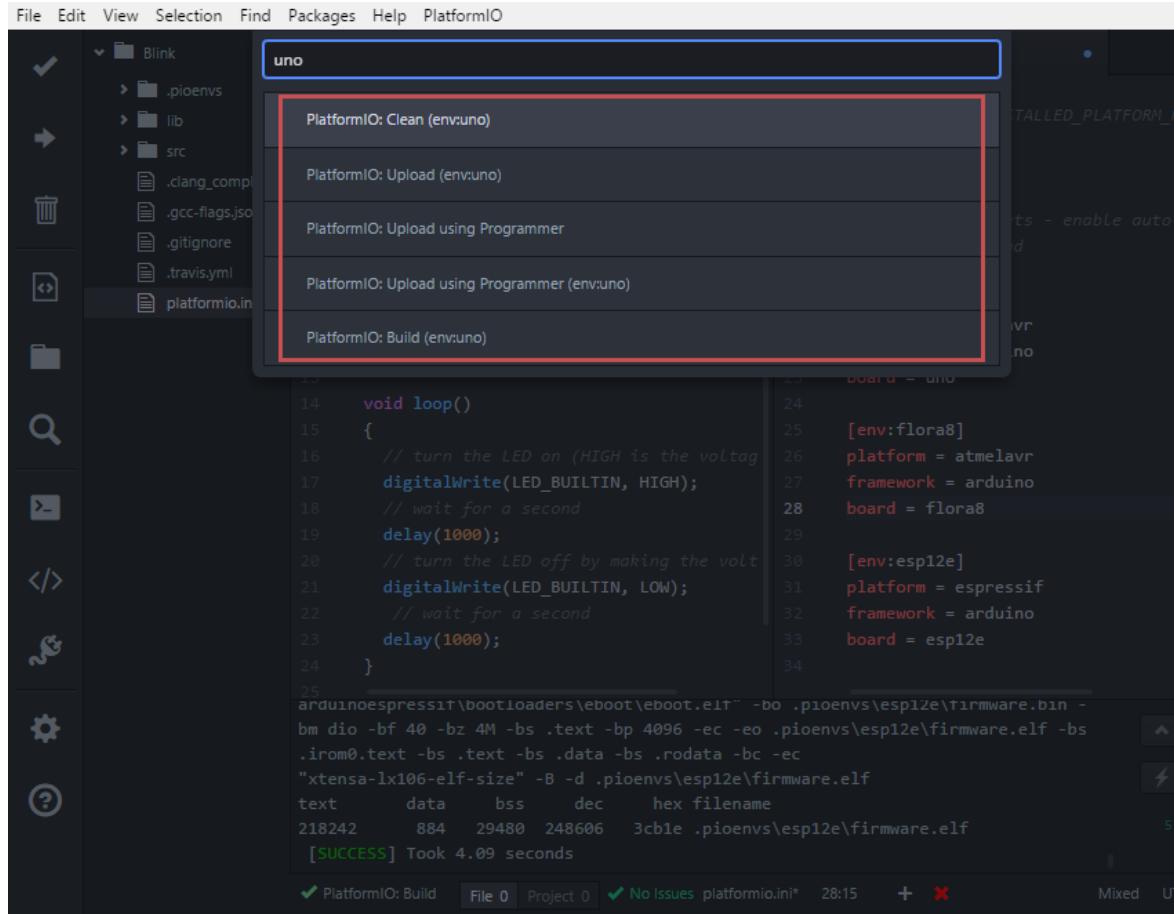
```

The bottom status bar also includes buttons for 'PIO Build' (+, X), file paths ('src/main.cpp'), and status indicators (LF, UTF-8, C++, 0 files).

To upload firmware to the board run Upload.

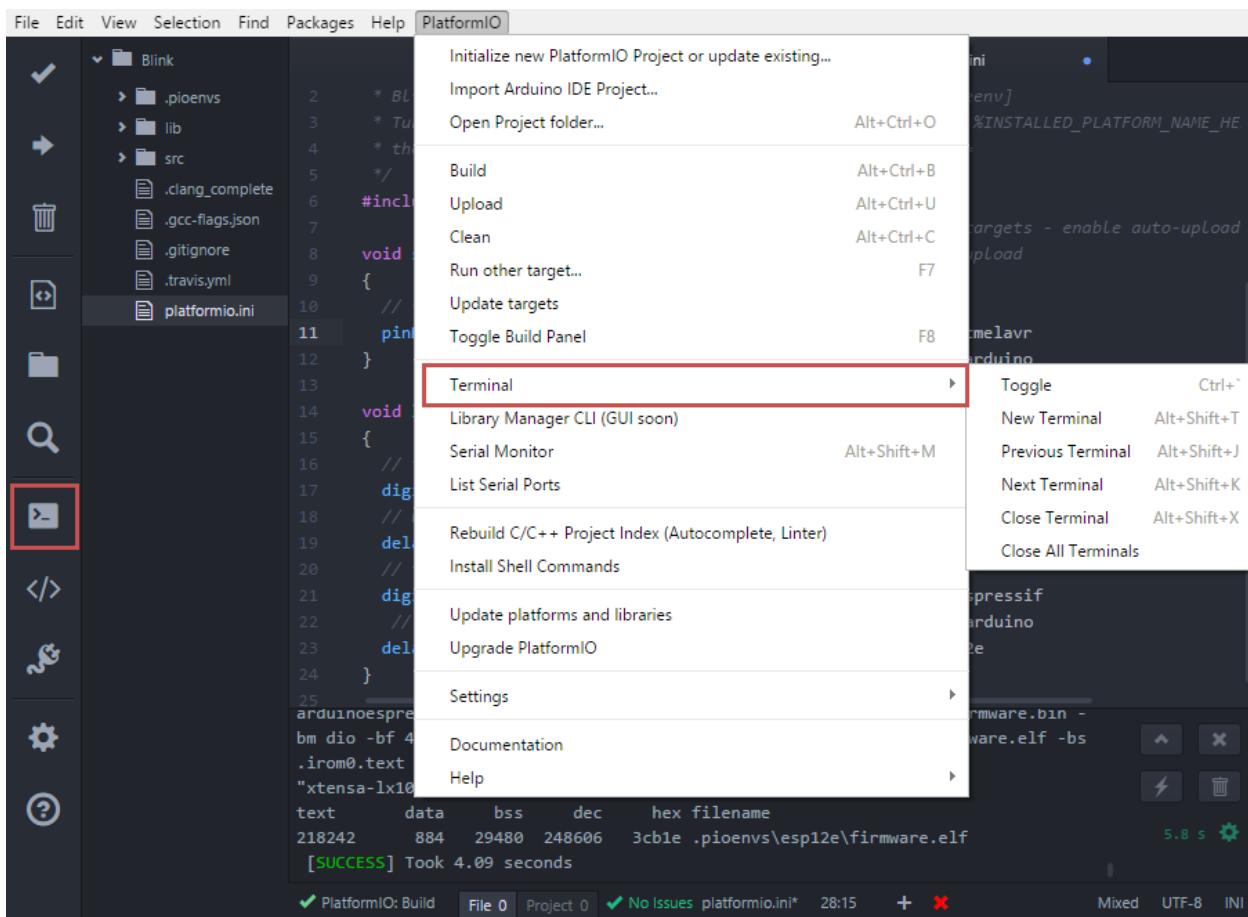
6. What is more, you can run specific target or process project environment using Menu: PlatformIO > Run other target... or call targets list from the status bar (bottom, left corner):



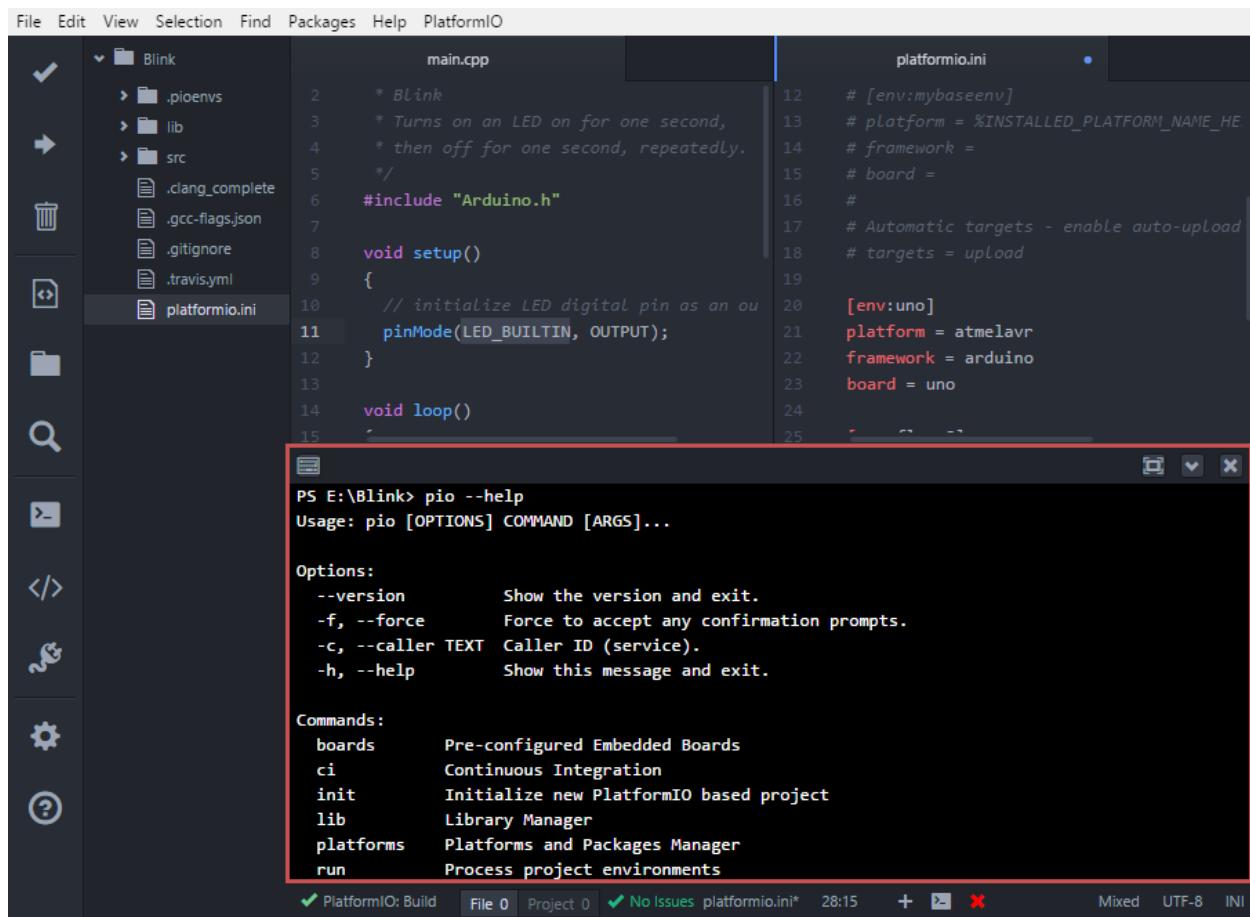


And select desired target:

7. To run built-in terminal interface choose Menu: PlatformIO > Terminal or press the corresponding icon in the PlatformIO toolbar:



It provides you fast access to all set of powerful *PlatformIO Core* CLI commands:



The screenshot shows the PlatformIO IDE interface. On the left is a toolbar with various icons. The main area has a file tree on the left showing a project named 'Blink' with files like .pioenvs, lib, src, .clang_complete, .gcc-flags.json, .gitignore, .travis.yml, and platformio.ini. The central pane displays the contents of 'main.cpp' and 'platformio.ini'. The right pane shows a terminal window with the output of the command 'pio --help', which provides usage information for the PlatformIO command-line interface.

```

main.cpp
2  * Blink
3  * Turns on an LED on for one second,
4  * then off for one second, repeatedly.
5  */
6  #include "Arduino.h"
7
8  void setup()
9  {
10     // initialize LED digital pin as an output
11     pinMode(LED_BUILTIN, OUTPUT);
12 }
13
14 void loop()
15

platformio.ini
12 # [env:mybaseenv]
13 # platform = %INSTALLED_PLATFORM_NAME_HE_
14 # framework =
15 # board =
16 #
17 # Automatic targets - enable auto-upload
18 # targets = upload
19
20 [env:uno]
21 platform = atmelavr
22 framework = arduino
23 board = uno
24
25

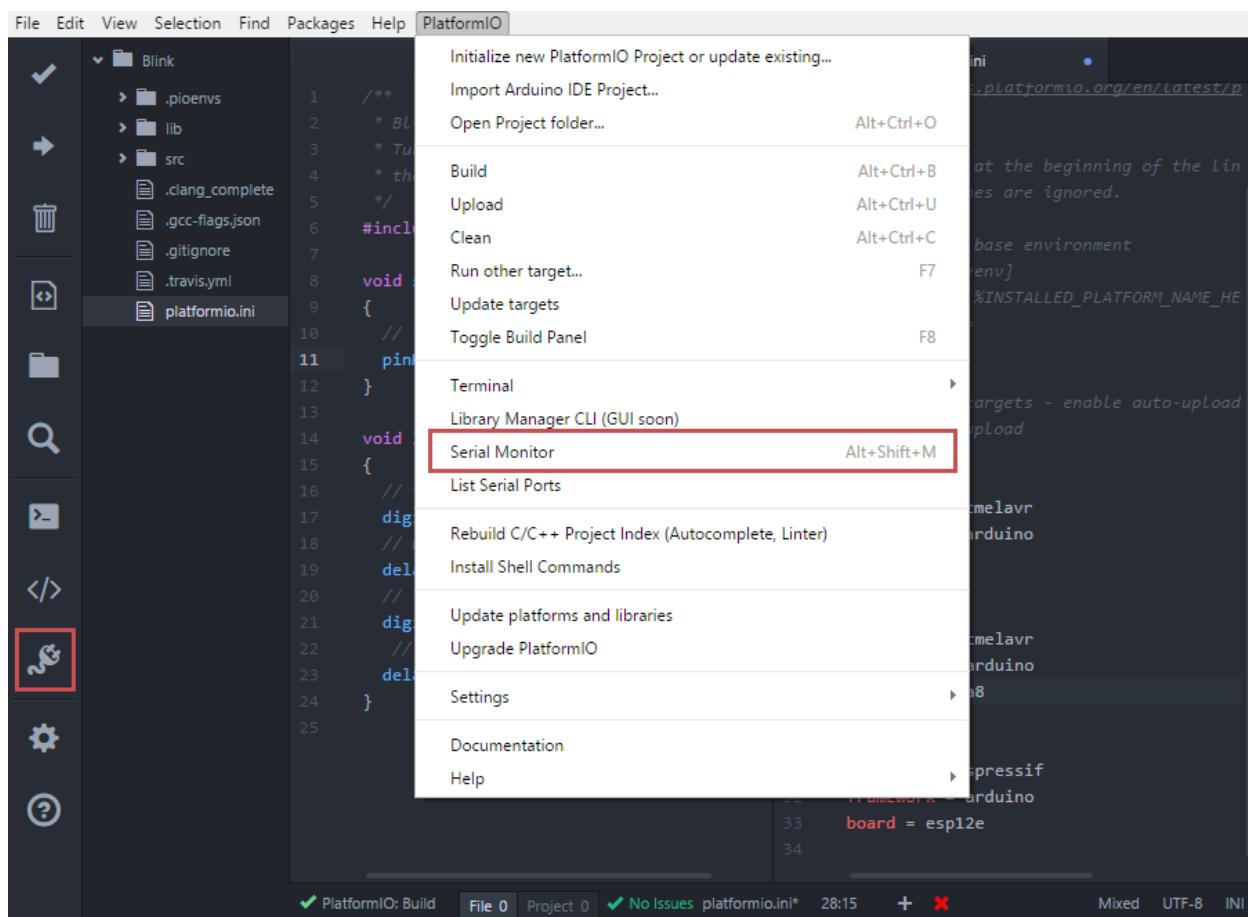
PS E:\Blink> pio --help
Usage: pio [OPTIONS] COMMAND [ARGS]...

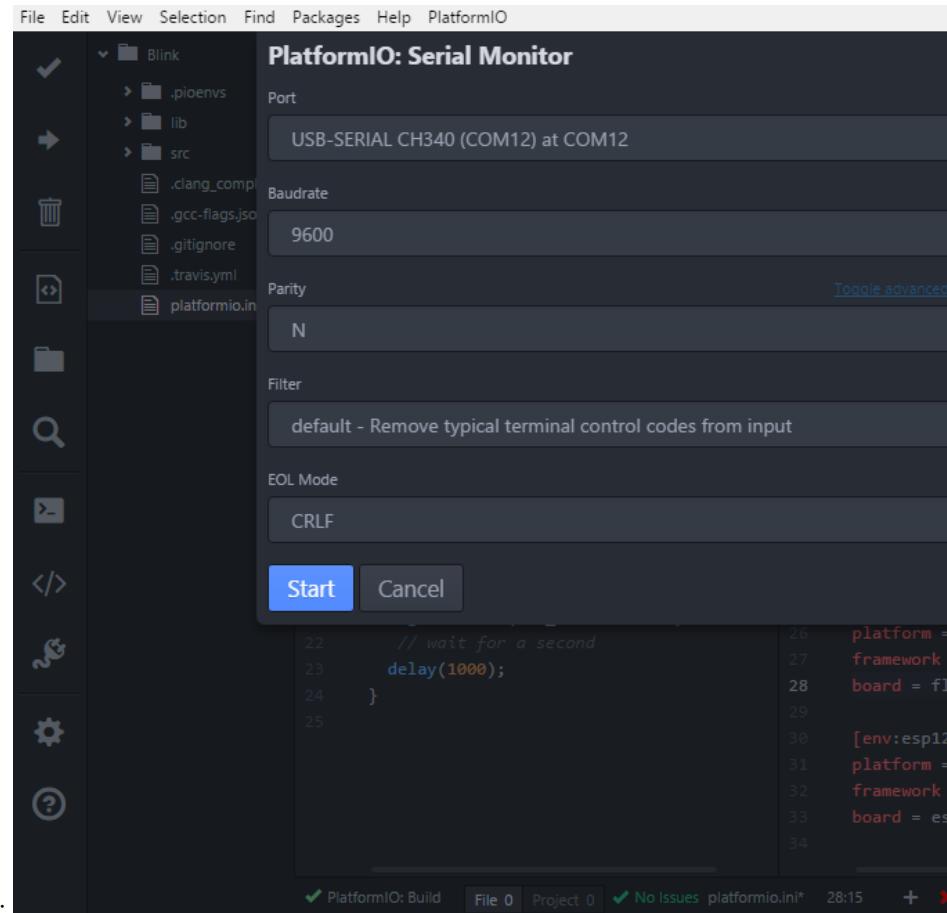
Options:
--version      Show the version and exit.
-f, --force    Force to accept any confirmation prompts.
-c, --caller TEXT Caller ID (service).
-h, --help     Show this message and exit.

Commands:
boards       Pre-configured Embedded Boards
ci           Continuous Integration
init         Initialize new PlatformIO based project
lib          Library Manager
platforms    Platforms and Packages Manager
run          Process project environments

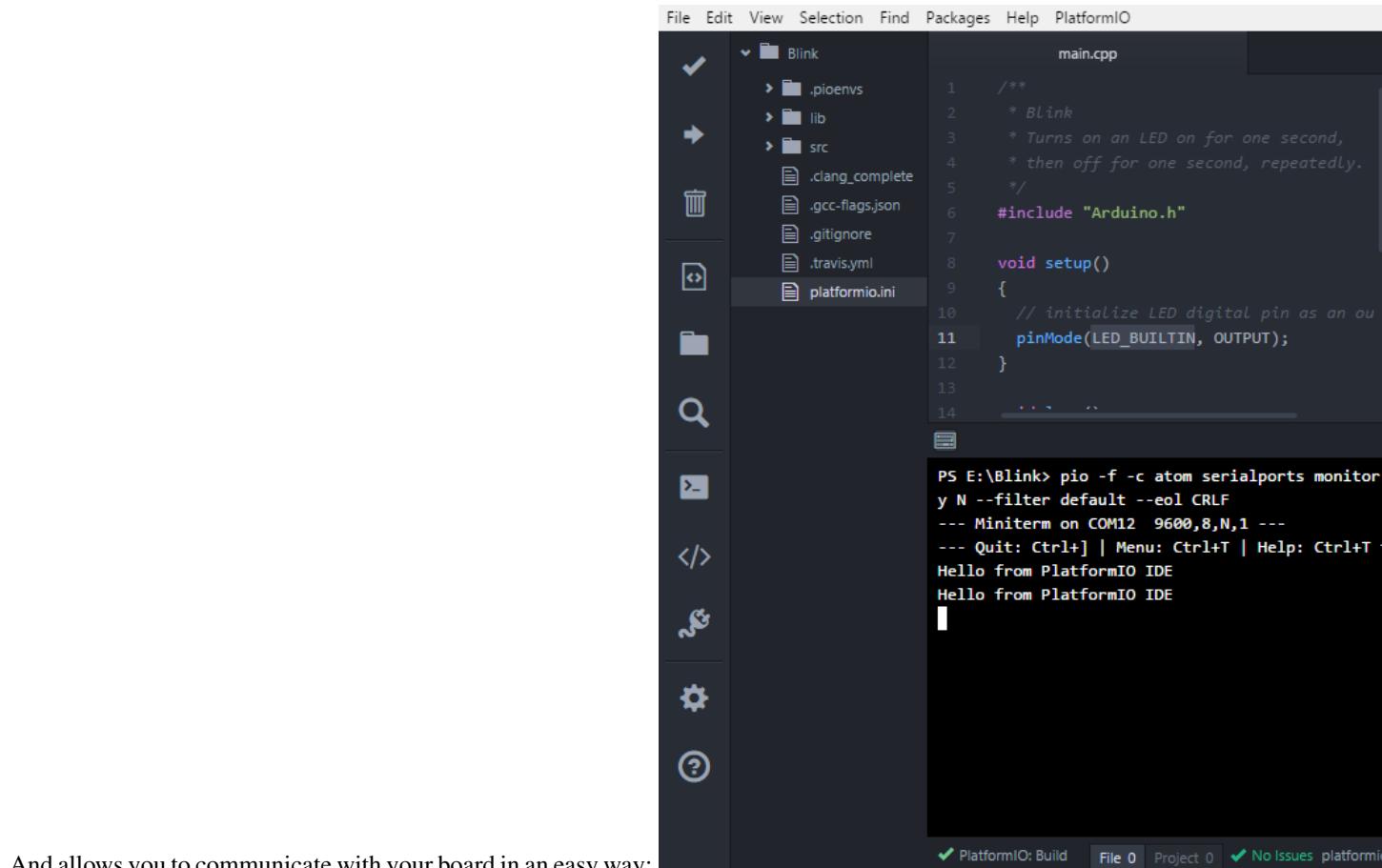
```

8. To run built-in “Serial Monitor” choose Menu: PlatformIO > Serial Monitor or press the corresponding icon in the PlatformIO toolbar:





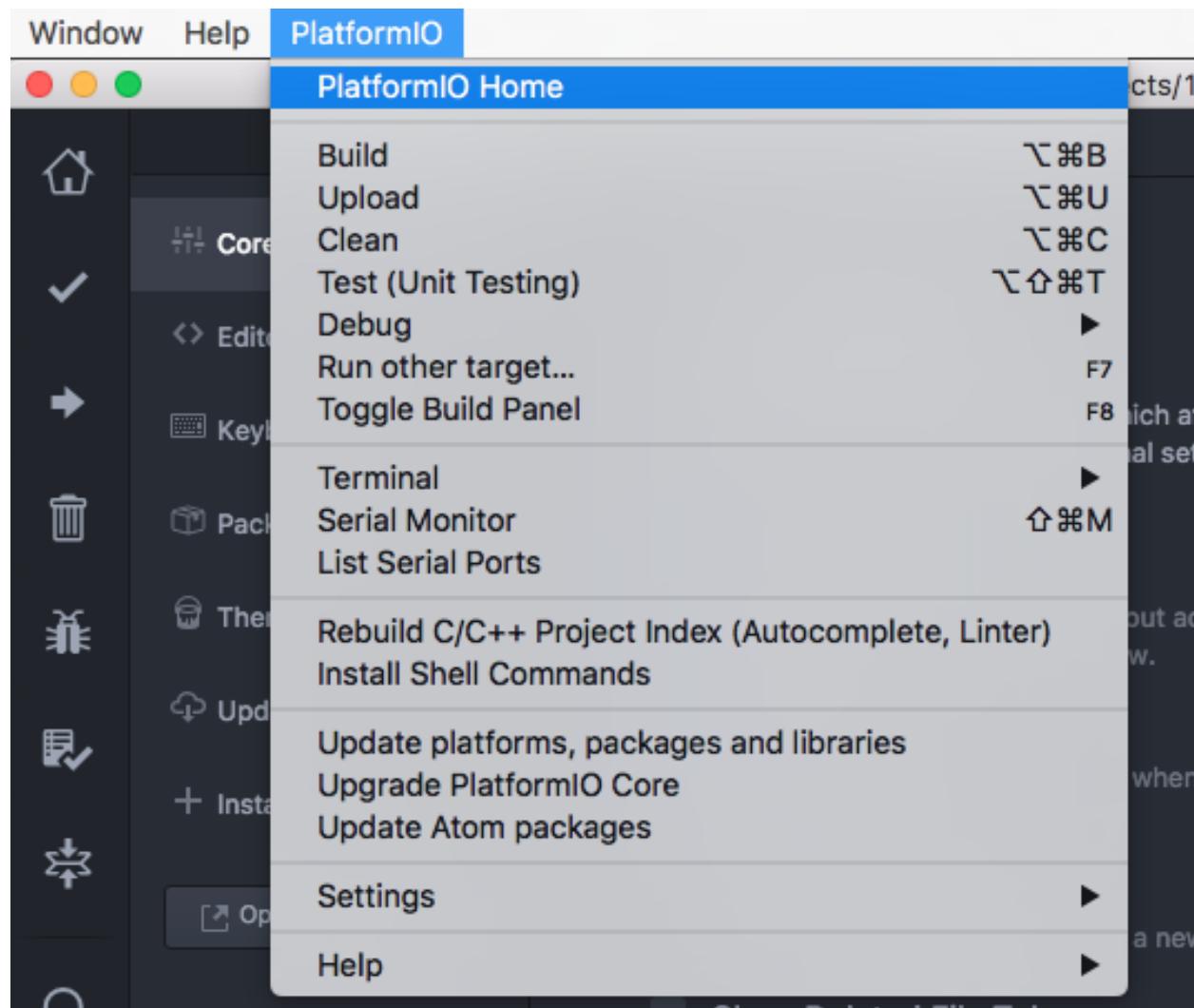
It has several settings to adjust your connection:



And allows you to communicate with your board in an easy way:

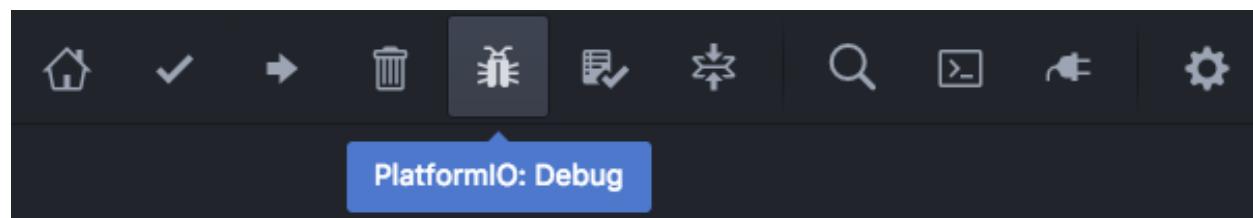
Menu item PlatformIO

platformio-ide package adds to Atom new menu item named **Menu: PlatformIO** (after **Menu: Help** item).



PlatformIO Toolbar

PlatformIO IDE Toolbar contains quick access buttons for the popular commands. Each button contains hint (delay mouse on it).



- [PlatformIO Home](#)
- PlatformIO: Build
- PlatformIO: Upload
- PlatformIO: Clean
- *PIO Unified Debugger*

- Run other target (Build environments, *PIO Unit Testing*)
- Toggle build panel
- ||
- Find in Project...
- PIO Terminal
- Serial Monitor
- ||
- Atom Settings

Building / Uploading / Targets

- cmd-alt-b / ctrl-alt-b / f9 builds project without auto-uploading.
- cmd-alt-u / ctrl-alt-u builds and uploads (if no errors).
- cmd-alt-c / ctrl-alt-c cleans compiled objects.
- cmd-alt-t / ctrl-alt-t / f7 run other targets (Upload using Programmer, Upload SPIFFS image, Update platforms and libraries).
- cmd-alt-g / ctrl-alt-g / f4 cycles through causes of build error.
- cmd-alt-h / ctrl-alt-h / shift-f4 goes to the first build error.
- cmd-alt-v / ctrl-alt-v / f8 toggles the build panel.
- escape terminates build / closes the build window.

More options Menu: PlatformIO > Settings > Build.

Intelligent Code Completion

PlatformIO IDE uses [clang](#) for the Intelligent Code Completion. To install it or check if it is already installed, please follow to step *II. Clang for Intelligent Code Completion* from Installation guide.

Warning: The libraries which are added/installed after initializing process will not be reflected in code linter. You need Menu: PlatformIO > Rebuild C/C++ Project Index (Autocomplete, Linter).

Smart Code Linter

PlatformIO IDE uses PlatformIO's pre-built GCC toolchains for Smart Code Linter and rapid professional development. The configuration data are located in `.gcc-flags.json`. This file will be automatically created and preconfigured when you initialize project using Menu: PlatformIO > Initialize new PlatformIO Project or update existing....

Warning: If some libraries are not visible in *PlatformIO IDE for Atom* and Code Completion or Code Linting does not work properly, please perform Menu: PlatformIO > Rebuild C/C++ Project Index (Autocomplete, Linter)

Install Shell Commands

Please navigate to PIO Core [Install Shell Commands](#).

Known issues

Smart Code Linter is disabled for Arduino files

Smart Code Linter is disabled by default for Arduino files (*.ino and .pde) because they are not valid C/C++ based source files:

1. Missing includes such as `#include <Arduino.h>`
2. Function declarations are omitted.

There are two solutions:

- [Convert Arduino file to C++ manually](#)
- [Force Arduino file as C++](#)

Convert Arduino file to C++ manually

Recommended! See [Convert Arduino file to C++ manually](#).

Force Arduino file as C++

To force Smart Code Linter to use Arduino files as C++ please

1. Open `.gcc-flags.json` file from the Initialized/Imported project and add `-x c++` flag at the beginning of the value of `gccDefaultCppFlags` field:

```
{  
    "execPath": "...",  
    "gccDefaultCFlags": "...",  
    "gccDefaultCppFlags": "-x c++ -fsyntax-only ...",  
    "gccErrorLimit": 15,  
    "gccIncludePaths": "...",  
    "gccSuppressWarnings": false  
}
```

2. Perform all steps from [Convert Arduino file to C++ manually](#) (without renaming to `.cpp`).

Warning: Please do not modify other flags here. They will be removed on a next “Project Rebuild C/C++ Index” stage. Please use `build_flags` for *Project Configuration File platformio.ini* instead.

Arch Linux: PlatformIO IDE Terminal issue

Please read this article [Installing PlatformIO on Arch Linux](#).

Frequently Asked Questions

Keep build panel visible

PlatformIO IDE hides build panel on success by default. Nevertheless, you can keep it visible all time. Please follow to Menu: PlatformIO > Settings > Build and set Panel Visibility to Keep Visible.

Key-bindings (toggle panel):

- cmd+alt+v - Mac OS X
- ctrl+alt+v - Windows/Linux

Automatically save on build

If you want automatically save all edited files when triggering a build, please follow to Menu: PlatformIO > Settings > Build and check Automatically save on build.

Jump to Declaration

Click on a function/include, press F3 and you will be taken directly to the declaration for that function.

Code Formatting

You need to install `atom-beautify` package and `C/C++ Uncrustify Code Beautifier`.

Uninstall Atom with PlatformIO IDE

Here's how to uninstall the PlatformIO IDE for multiple OS.

See [Uninstall PIO Core and dependent packages](#), if you do not need it in a system.

Windows

1. Uninstall Atom using “Start > Control Panel > Programs and Features > Uninstall”
2. Remove `C:\Users\<user name>\.atom` folder (settings, packages, etc...)
3. Remove `C:\Users\<user name>\AppData\Local\atom` folder (application itself)
4. Remove `C:\Users\<user name>\AppData\Roaming\Atom` folder (cache, etc.)
5. Remove registry records using `regedit`:
 - `HKEY_CLASSES_ROOT\Directory\Background\shell`
 - `HKEY_CLASSES_ROOT\Directory\shell`
 - `HKEY_CLASSES_ROOT*\shell`

macOS

Run these commands in system Terminal

```
rm -rf ~/.atom
rm /usr/local/bin/atom
rm /usr/local/bin/apm
rm -rf /Applications/Atom.app
rm ~/Library/Preferences/com.github.atom.plist
rm ~/Library/Application\ Support/com.github.atom.ShipIt
rm -rf ~/Library/Application\ Support/Atom
rm -rf ~/Library/Saved\ Application\ State/com.github.atom.savedState
rm -rf ~/Library/Caches/com.github.atom
rm -rf ~/Library/Caches/Atom
```

Linux

Run these commands in system Terminal

```
rm /usr/local/bin/atom
rm /usr/local/bin/apm
rm -rf ~/.atom
rm -rf ~/.config/Atom-Shell
rm -rf /usr/local/share/atom/
```

Articles / Manuals

- Mar, 31, 2017 - **Robin Reiter** - A little guide to PlatformIO. As an Arduino developer, you may want to check that out! ([video review](#))
- Dec 13, 2016 - **Dr. Patrick Mineault** - Multi-Arduino projects with PlatformIO
- Nov 10, 2016 - **PiGreek** - PlatformIO the new Arduino IDE ?!
- Aug 18, 2016 - **Primal Cortex** - Installing PlatformIO on Arch Linux
- Jul 26, 2016 - **Embedded Systems Laboratory** - PlatformIO IDE Arduino ESP8266 (Get started with PlatformIO IDE for Arduino board and ESP8266, Thai)
- May 30, 2016 - **Ron Moerman** - IoT Development with PlatformIO
- May 01, 2016 - **Pedro Minatel** - PlatformIO – Uma alternativa ao Arduino IDE (PlatformIO - An alternative to the Arduino IDE, Portuguese)
- Apr 23, 2016 - **AI Williams** - Hackaday: Atomic Arduino (and Other) Development
- Apr 16, 2016 - **Sathittham Sangthong** - [PlatformIO] PlatformIO Arduino IDE (Let's play together with PlatformIO IDE [alternative to Arduino IDE], Thai)
- Apr 11, 2016 - **Matjaz Trcek** - Top 5 Arduino integrated development environments
- Apr 06, 2016 - **Aleks** - PlatformIO ausprobiert (Tried PlatformIO, German)
- Apr 02, 2016 - **Diego Pinto** - Você tem coragem de abandonar a IDE do Arduino? PlatformIO + Atom (Do you dare to leave the Arduino IDE? PlatformIO + Atom, Portuguese)
- Mar 30, 2016 - **Brandon Cannaday** - Getting Started with PlatformIO and ESP8266 NodeMcu

- Mar 12, 2016 - **Peter Marks** - PlatformIO, the Arduino IDE for programmers
- Mar 05, 2016 - **brichacek.net** - PlatformIO – otevřený ekosystém pro vývoj IoT (PlatformIO – an open source ecosystem for IoT development, Czech)
- Mar 04, 2016 - **Ricardo Vega** - Programa tu Arduino desde Atom (Program your Arduino from Atom, Spanish)
- Feb 28, 2016 - **Alex Bloggt** - PlatformIO vorgestellt (Introduction to PlatformIO IDE, German)
- Feb 25, 2016 - **NutDIY** - PlatformIO Blink On Nodemcu Dev Kit V1.0 (Thai)

See a full list with [Articles about us](#).

Changelog

Please visit [releases page](#).

1.18.2 Cloud IDE

Cloud9

Cloud9 combines a powerful online code editor with a full Ubuntu workspace in the cloud. Workspaces are powered by Docker Ubuntu containers that give you full freedom over your environment, including sudo rights. Do a git push, compile SASS, see server output, and Run apps easily with the built-in Terminal and Runners.

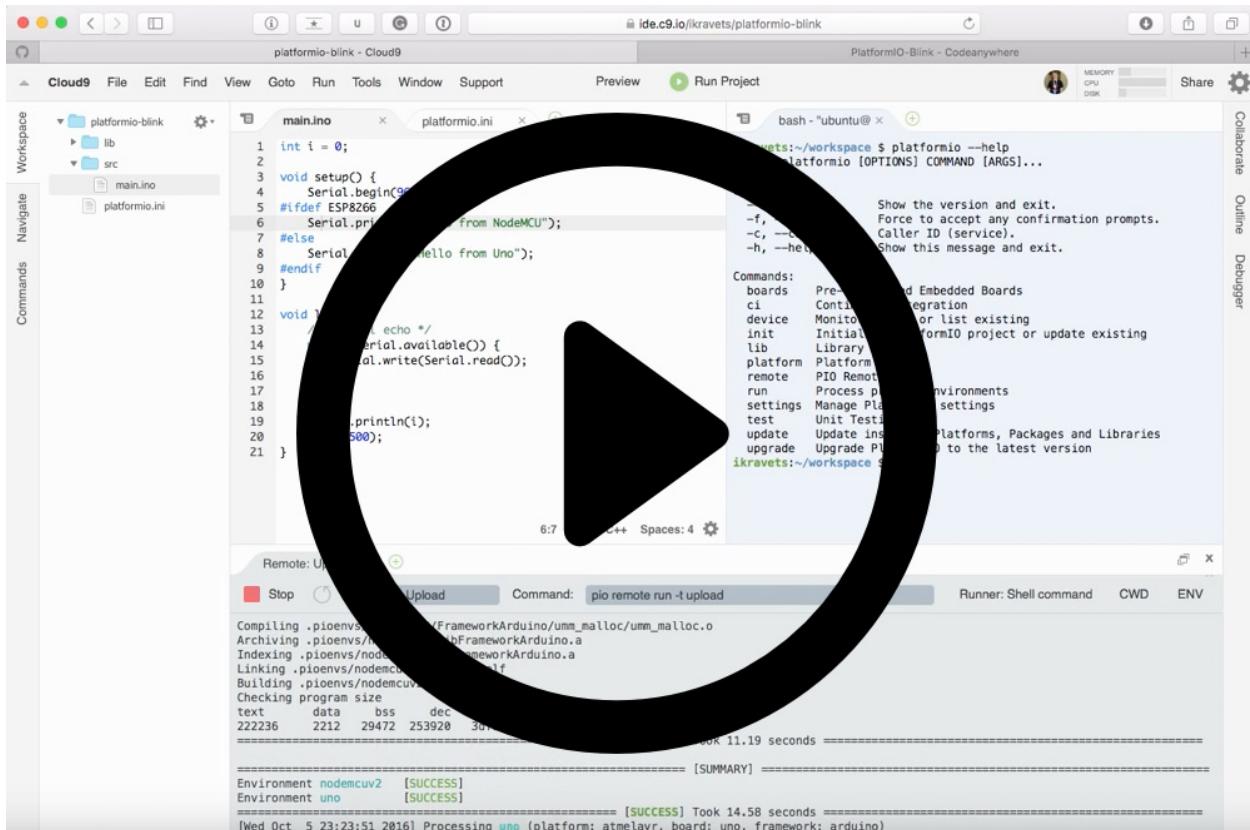
Contents

- [Cloud9](#)
 - [Demo](#)
 - [Integration](#)
 - [Quick Start](#)
 - [PlatformIO Build System](#)
 - [Remote Device Manager](#)
 - [Remote Firmware Uploading](#)
 - [Remote Serial Port Monitor](#)
 - [Multi-Project workspace](#)

Note:

1. Please make sure to read [PIO Remote](#) guide first.
 2. You need [PIO Account](#) if you don't have it. Registration is FREE.
 3. You should have a running [PIO Remote Agent](#) on a remote machine where hardware devices are connected physically or accessible for the remote operations. See [PIO Remote Quick Start](#) for details.
-

Demo



Integration

1. Sign in to Cloud9. A registration is FREE and gives you for FREE 1 private workspace (where you can host multiple PlatformIO Projects) and unlimited public workspaces.
2. Create a new workspace using **Blank** template

Create a new workspace

Workspace name

Description

[Hosted workspace](#)

[Clone workspace](#)

[Remote SSH workspace](#)

[Salesforce](#)

 **Private**

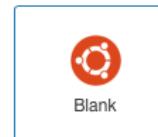
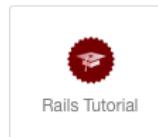
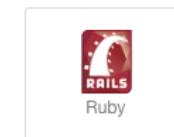
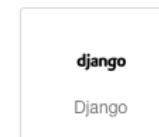
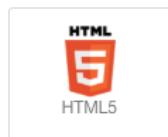
This is a workspace for your eyes only

 **Public**

This will create a workspace for everybody to see

Clone from Git or Mercurial URL (optional)

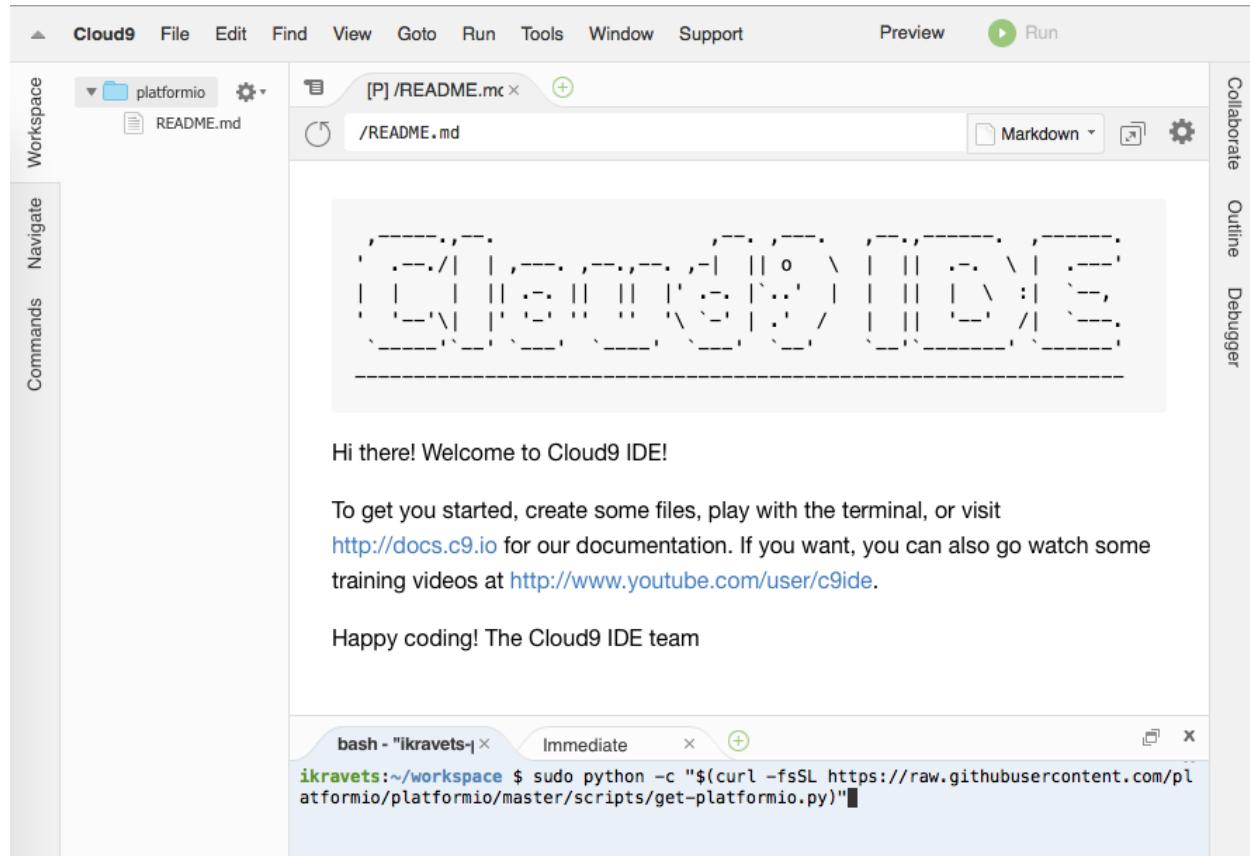
Choose a template



[Create workspace](#)

3. Install *PlatformIO Core* using Cloud IDE Terminal. Paste a next command

```
sudo python -c "$(curl -fSSL https://raw.githubusercontent.com/platformio/platformio/develop/scripts/get-platformio.py)"
```



4. Log in to *PIO Account* using *platformio account login* command.

Quick Start

Let's create our first PlatformIO-based Cloud9 Project

1. Initialize new PlatformIO-based Project. Run a next command in Cloud IDE Terminal:

```
platformio init --board <ID>
# initialize project for Arduino Uno
platformio init --board uno
```

To get board ID please use *platformio boards* command or *Embedded Boards Explorer*.

2. Create new source file named `main.cpp` in `src` folder using Project Tree (left side). Please make right click on `src` folder, then “New File” and insert a next content:

```
#include <Arduino.h>

int i = 0;

void setup() {
    Serial.begin(9600);
    Serial.println("Hello Cloud9!");
}
```

(continues on next page)

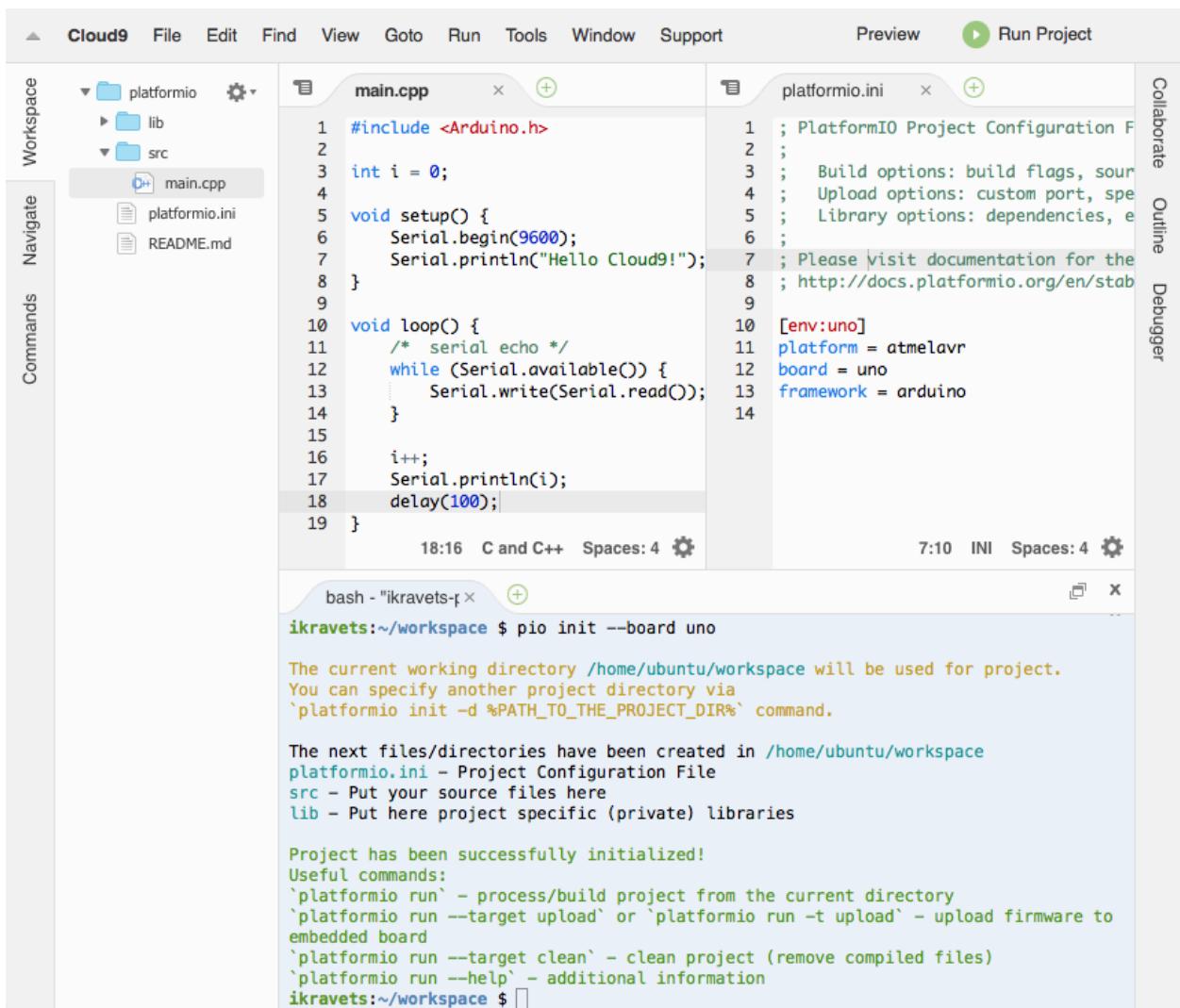
(continued from previous page)

```

void loop() {
    /* serial echo */
    while (Serial.available()) {
        Serial.write(Serial.read());
    }

    i++;
    Serial.println(i);
    delay(100);
}

```



3. If you prefer to work with *PlatformIO Core* CLI, then you can process project using Cloud IDE Terminal and the next commands:

- *platformio run* - build project locally (using Cloud IDE's virtual machine)
- *pio run -t clean* - clean project
- *pio remote run -t upload* - upload firmware (program) to a remote device
- *platformio remote device list* - list available remote devices

- *platformio remote device monitor* - Remote Serial Port Monitor

If you are interested in better integration with Cloud9 and GUI, please read guide below where we will explain how to create custom Build System for PlatformIO and own Runners.

PlatformIO Build System

Cloud9 allows to create own build system and use hotkey or command (Menu: Run > Build) to build a project.

Let's create PlatformIO Build System that will be used for C/C++/H/INO/PDE files by default. Please click on Menu : Run > Build System > New Build System and replace all content with the next:

```
{  
    "cmd" : ["pio", "run", "-d", "$file"],  
    "info" : "Building $project_path/$file_name",  
    "selector": "^.*\\.(cpp|c|h|hpp|S|ini|ino|pde)$"  
}
```

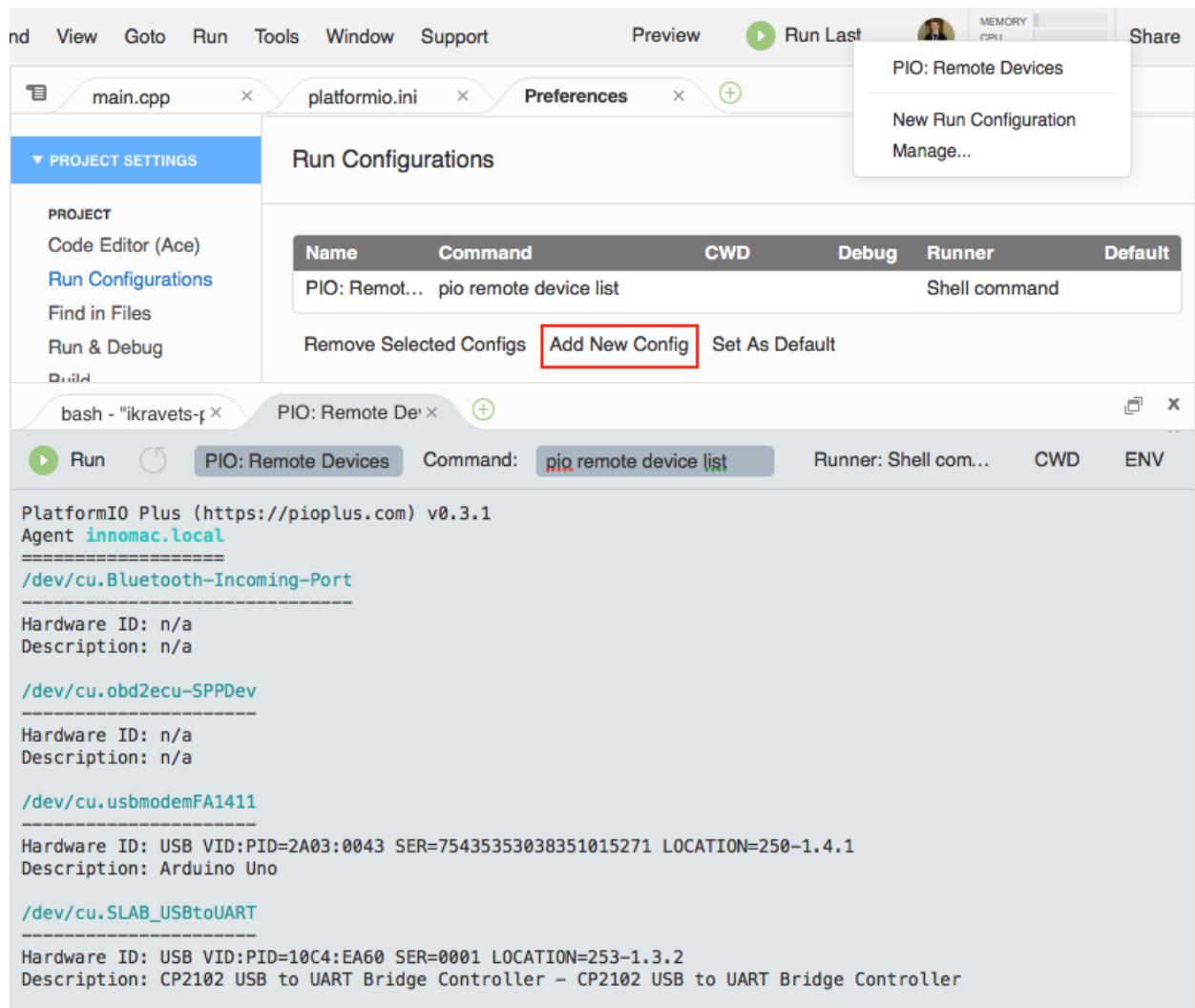
Save new Build System and give a name PIOBuilder. Now, you can select it as default Build System using Menu : Run > Build System > PIOBuilder.

Remote Device Manager

Remote Device Manager works in pair with *PIO Remote*. You can list remote devices that are connected to host machine where *PIO Remote Agent* is started or are visible for it.

Let's create New Run Configuration (shortcut) that will be used for Remote Device Manager. Please click on Menu : Run > Run Configurations > Manage..., then “Add New Config” and specify the next values:

- **First Blank Input:** a name of runner. Please set it to “PIO: Remote Devices”
- **Command:** set to `pio remote device list`
- **Runner:** set to “Shell command”



Remote Firmware Uploading

Remote Firmware Uploading works in pair with [PIO Remote](#). You can deploy firmware (program) to any devices which are visible for [PIO Remote Agent](#).

Let's create New Run Configuration (shortcut) that will be used for Remote Firmware Uploading. Please click on Menu: Run > Run Configurations > Manage..., then "Add New Config" and specify the next values:

- **First Blank Input:** a name of runner. Please set it to "PIO: Remote Upload"
- **Command:** set to pio remote run -t upload
- **Runner:** set to "Shell command"

The screenshot shows the PlatformIO IDE interface. At the top, there's a menu bar with File, View, Goto, Run, Tools, Window, Support, Preview, and a Run Project button. Below the menu is a toolbar with tabs for main.cpp, platformio.ini, Preferences, and a plus sign. A context menu is open over the Run Project button, with options like PIO: Remote Devices and PIO: Remote Upload selected.

The main area has a sidebar titled 'PROJECT SETTINGS' with sections for PROJECT (Code Editor (Ace), Run Configurations, Find in Files, Run & Debug, Build), and a central 'Run Configurations' section. This section contains a table:

Name	Command	CWD	Debug	Runner	Default
PIO: Remote Devices	pio remote device list			Shell command	
PIO: Remote Upload	pio remote run -t upload			Shell command	true

Buttons at the bottom of this section include Remove Selected Configs, Add New Config (which is highlighted with a red box), and Set As Default.

Below the configurations is a terminal window titled 'PIO: Remote Upd'. It shows the output of a remote upload process:

```

bash - "ikravets-p ~" ~
PIO: Remote Upd x
Run PIO: Remote Upload Command: pio remote run -t upload Runner: Shell com... CWD ENV
PlatformIO Plus (https://pioplus.com) v0.3.1
Building project locally
[Sat Oct 29 22:35:48 2016] Processing uno (platform: atmelavr, board: uno, framework: arduino)

Verbose mode can be enabled via `--verbose` option
Collected 25 compatible libraries
Looking for dependencies...
Project does not have dependencies
Checking program size
text      data      bss      dec      hex filename
2324        48     168    2540   9ec .pioenvs/uno/firmware.elf
===== [SUCCESS] Took 0.46 seconds =====
Uploading firmware remotely
[Sun Oct 30 01:35:49 2016] Processing uno (platform: atmelavr, board: uno, framework: arduino)

Verbose mode can be enabled via `--verbose` option
Looking for upload port...
Auto-detected: /dev/cu.usbmodemFA1411
Uploading .pioenvs/uno/firmware.hex

avrduude: AVR device initialized and ready to accept instructions
Reading | ##### | 100% 0.00s
avrduude: Device signature = 0x1e950f
avrduude: reading input file ".pioenvs/uno/firmware.hex"
avrduude: writing flash (2372 bytes):
Writing | ##### | 100% 0.39s
avrduude: 2372 bytes of flash written
avrduude: verifying flash memory against .pioenvs/uno/firmware.hex:
avrduude: load data flash data from input file .pioenvs/uno/firmware.hex:
avrduude: input file .pioenvs/uno/firmware.hex contains 2372 bytes
avrduude: reading on-chip flash data:
Reading | ##### | 100% 0.31s
avrduude: verifying ...
avrduude: 2372 bytes of flash verified

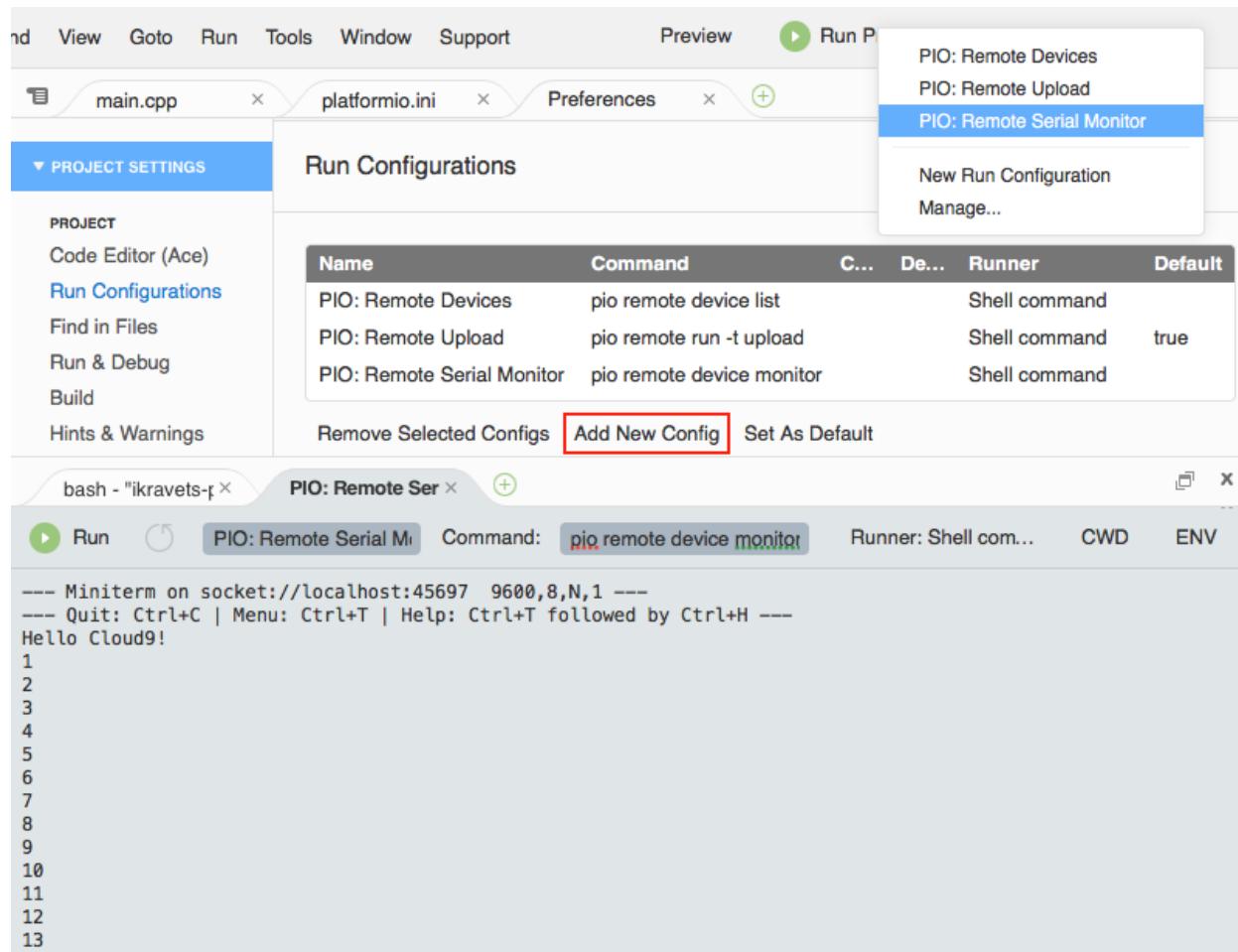
avrduude done. Thank you.
===== [SUCCESS] Took 3.19 seconds =====
  
```

Remote Serial Port Monitor

Remote Serial Port Monitor works in pair with [PIO Remote](#). You can read or send data to any device that is connected to host machine where [PIO Remote Agent](#) is started. To list active agents please use this command `platformio remote agent list`.

Let's create New Run Configuration (shortcut) that will be used for Remote Serial Port Monitor. Please click on Menu: Run > Run Configurations > Manage..., then "Add New Config" and specify the next values:

- **First Blank Input:** a name of runner. Please set it to "PIO: Remote Serial Monitor"
- **Command:** set to `pio remote device monitor`
- **Runner:** set to "Shell command"



Multi-Project workspace

You can have multiple PlatformIO-based Projects in the same workspace. We recommend a next folders structure:



(continued from previous page)



In this case, you need to create 2 “New Run Configuration” for *Remote Firmware Uploading* with using the next **commands**:

- `pio remote run --project-dir project-A -t upload` for Project-A
- `pio remote run -d project-B -t upload` for Project-B

See documentation for `platformio remote run --project-dir` option.

Codeanywhere

[Codeanywhere](#) is a Cross Platform Cloud IDE and it has all the features of Desktop IDE but with additional features only a cloud application can give you! Codeanywhere is very flexible and you can set up your workflow any way you want it. The elegant development environment will let you focus on building great applications quicker. All the features you will need for any coding task are built into Codeanywhere, making development more productive and fun.

Contents

- *Codeanywhere*
 - *Demo*
 - *Integration*
 - *Quick Start*
 - *Run Button*
 - *Remote Device Manager*
 - *Remote Firmware Uploading*
 - *Remote Serial Port Monitor*
 - *Multi-Project workspace*

Note:

1. Please make sure to read [PIO Remote](#) guide first.
2. You need [PIO Account](#) if you don't have it. Registration is FREE.
3. You should have a running [PIO Remote Agent](#) on a remote machine where hardware devices are connected physically or accessible for the remote operations. See [PIO Remote Quick Start](#) for details.

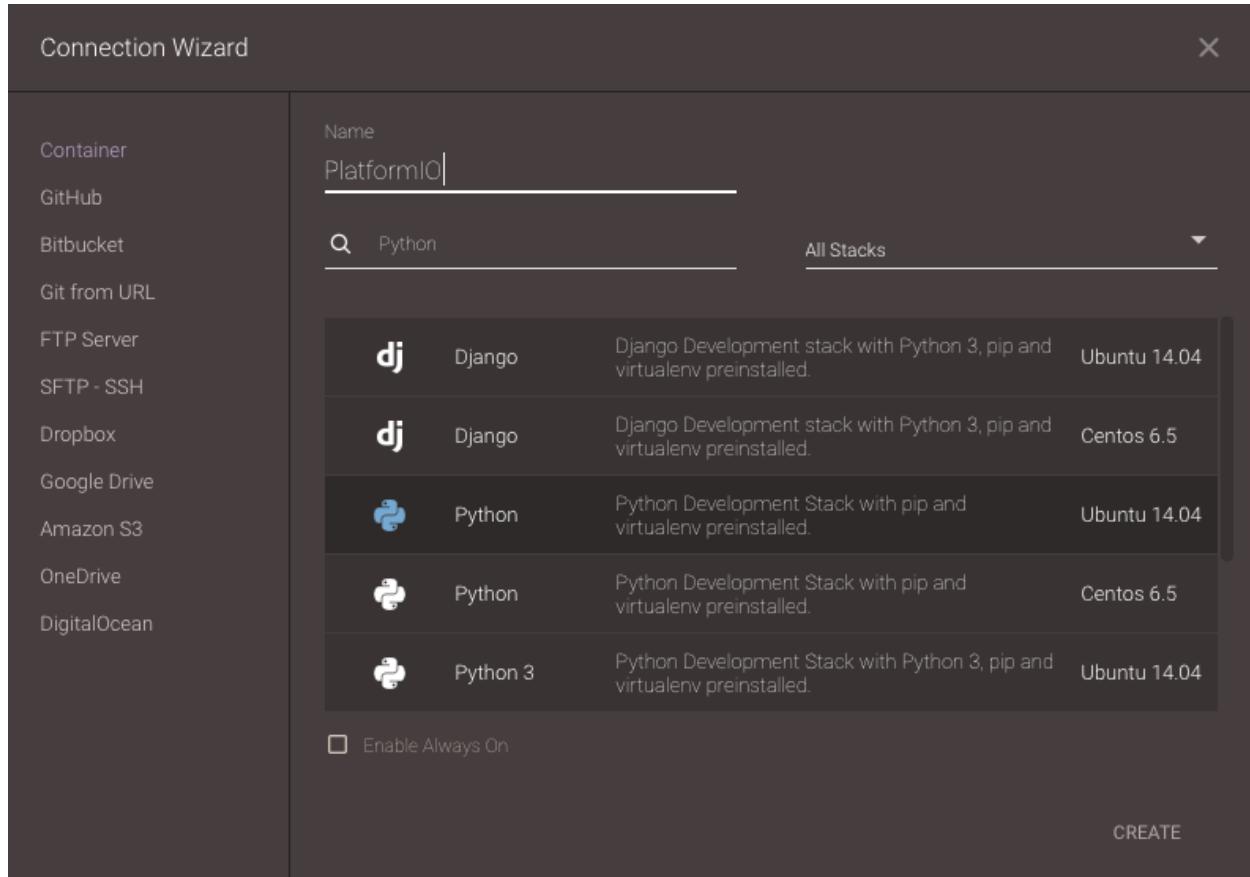
Demo

The screenshot shows the PlatformIO IDE integrated into the Codeanywhere environment. The interface includes:

- File Menu:** File, Edit, Find, Goto, View, Preferences, Help.
- Sidebar:** Shows the project structure: platformIO (lib, src), main.cpp, and platformio.ini.
- Code Editors:**
 - main.cpp:** Contains C++ code for a serial echo application.
 - platformio.ini:** Project configuration file.
- Terminal:** Shows the command `pio init -b uno` being run, followed by the output of the initialization process.

Integration

1. Sign in to Codeanywhere. A registration is FREE and gives you unlimited private projects within the one Container.
2. Open Dashboard Projects
3. Create a new Project and open it. In Connection Wizard create new Container:
 - **Name** set to “PlatformIO”
 - **Stack** search for Python stack (not Python3) that is based on Ubuntu OS.
 - Click on “Create” button.



4. Open **SSH-Terminal** tab (right click on Container (PlatformIO) > SSH Terminal) and install *PlatformIO Core* using a next command

```
sudo python -c "$(curl -fsSL https://raw.githubusercontent.com/platformio/platformio/develop/scripts/get-platformio.py)"
```

```

platformio init --board <ID>
# initialize project for Arduino Uno
platformio init --board uno

```

```

PlatformIO -> PlatformIO
  □ Container Info  * PlatformIO  x  * PlatformIO  x
  □ PlatformIO  > SSH Terminal
    ■ Turn Off
    Q Restart
    E Enable Always On
    I Info
    C Config
    Create Custom Stack
    R Rename
    ▶ Run
    ⌂ Refresh
    Create File
    Create Folder
    Download
    Upload
    Share
    X Destroy
  lling Python Package Manager ...
  nt already up-to-date: pip in /usr/local/lib/python2.7/dist-packages
  lling PlatformIO and dependencies ...
  g https://github.com/platformio/platformio/archive/develop.zip
  ding https://github.com/platformio/platformio/archive/develop.zip (13.0MB)
  nt already up-to-date: bottle<0.13 in /usr/local/lib/python2.7/dist-packages (from
  o==3.2.0a11)
  nt already up-to-date: click<6,>=5 in /usr/local/lib/python2.7/dist-packages (from
  o==3.2.0a11)
  nt already up-to-date: lockfile<0.13,>=0.9.1 in /usr/local/lib/python2.7/dist-pac
  platformio==3.2.0a11)
  nt already up-to-date: requests<3,>=2.4.0 in /usr/local/lib/python2.7/dist-packages
  atformio==3.2.0a11)
  nt already up-to-date: semantic_version>=2.5.0 in /usr/local/lib/python2.7/dist-pac
  om platformio==3.2.0a11)
  nt already up-to-date: colorama in /usr/local/lib/python2.7/dist-packages (from pla
  3.2.0a11)
  nt already up-to-date: pyserial<4,>=3 in /usr/local/lib/python2.7/dist-packages (fr
  rmio==3.2.0a11)
  g collected packages: platformio
  xisting installation: platformio 3.2.0a11
  talling platformio-3.2.0a11:
  cessfully uninstalled platformio-3.2.0a11
  setup.py install for platformio: started
  ng setup.py install for platformio: finished with status 'done'
  Successfully installed platformio-3.2.0a11
  [SUCCESS]
  ==> Installation process has been successfully FINISHED! <==

```

5. Log in to *PIO Account* using *platformio account login* command.

Quick Start

Let's create our first PlatformIO-based Codeanywhere Project

1. Initialize new PlatformIO-based Project. Run a next command in a Cloud IDE SSH Terminal:

```

platformio init --board <ID>
# initialize project for Arduino Uno
platformio init --board uno

```

To get board ID please use *platformio boards* command or *Embedded Boards Explorer*.

If you do not see created project, please refresh Project Tree using right-click on Container Name (PlatformIO) > Refresh.

2. Create new source file named `main.cpp` in `src` folder using Project Tree (left side). Please make right click on `src` folder, then "Create File" and insert a next content:

```

#include <Arduino.h>

int i = 0;

void setup() {

```

(continues on next page)

(continued from previous page)

```

Serial.begin(9600);
Serial.println("Hello Codeanywhere!");
}

void loop() {
    /* serial echo */
    while (Serial.available()) {
        Serial.write(Serial.read());
    }

    i++;
    Serial.println(i);
    delay(100);
}

```

The screenshot shows the PlatformIO Cloud IDE interface. At the top is a menu bar with File, Edit, Find, Goto, View, Preferences, and Help. Below the menu is a toolbar with various icons. The main area has three tabs: 'platformio' (file browser), 'main.cpp' (code editor), and 'platformio.ini' (configuration editor). The 'main.cpp' tab contains the C++ code shown above. The 'platformio.ini' tab contains the configuration file:

```

1 ; PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, ...
4 ; Upload options: custom port, ...
5 ; Library options: dependencies, ...
6 ;
7 ; Please visit documentation for ...
8 ; http://docs.platformio.org/en...
9
10; [env:uno]
11platform = atmelavr
12board = uno
13framework = arduino
14

```

Below the tabs is a terminal window with the following output:

```

Last login: Thu Nov  3 10:32:35 2016 from 54.186.244.104
cabox@box-codeanywhere:~/workspace$ pio init -b uno

The current working directory /home/cabox/workspace will be used for project.
You can specify another project directory via
`platformio init -d %PATH_TO_THE_PROJECT_DIR%` command.

The next files/directories have been created in /home/cabox/workspace
platformio.ini - Project Configuration File
src - Put your source files here
lib - Put here project specific (private) libraries

Project has been successfully initialized!
Useful commands:
`platformio run` - process/build project from the current directory
`platformio run --target upload` or `platformio run -t upload` - upload firmware to embedded board
`platformio run --target clean` - clean project (remove compiled files)
`platformio run --help` - additional information
cabox@box-codeanywhere:~/workspace$ █

```

3. If you prefer to work with *PlatformIO Core* CLI, then you can process project using Cloud IDE SSH Terminal and the next commands:

- *platformio run* - build project locally (using Cloud IDE's virtual machine)
 - *pio run -t clean* - clean project
 - *pio remote run -t upload* - upload firmware (program) to a remote device
 - *platformio remote device list* - list available remote devices
 - *platformio remote device monitor* - Remote Serial Port Monitor
4. We recommend to hide “Hidden Files”. You can do that via Cloud IDE Menu: View > Show Hidden Files.

Run Button

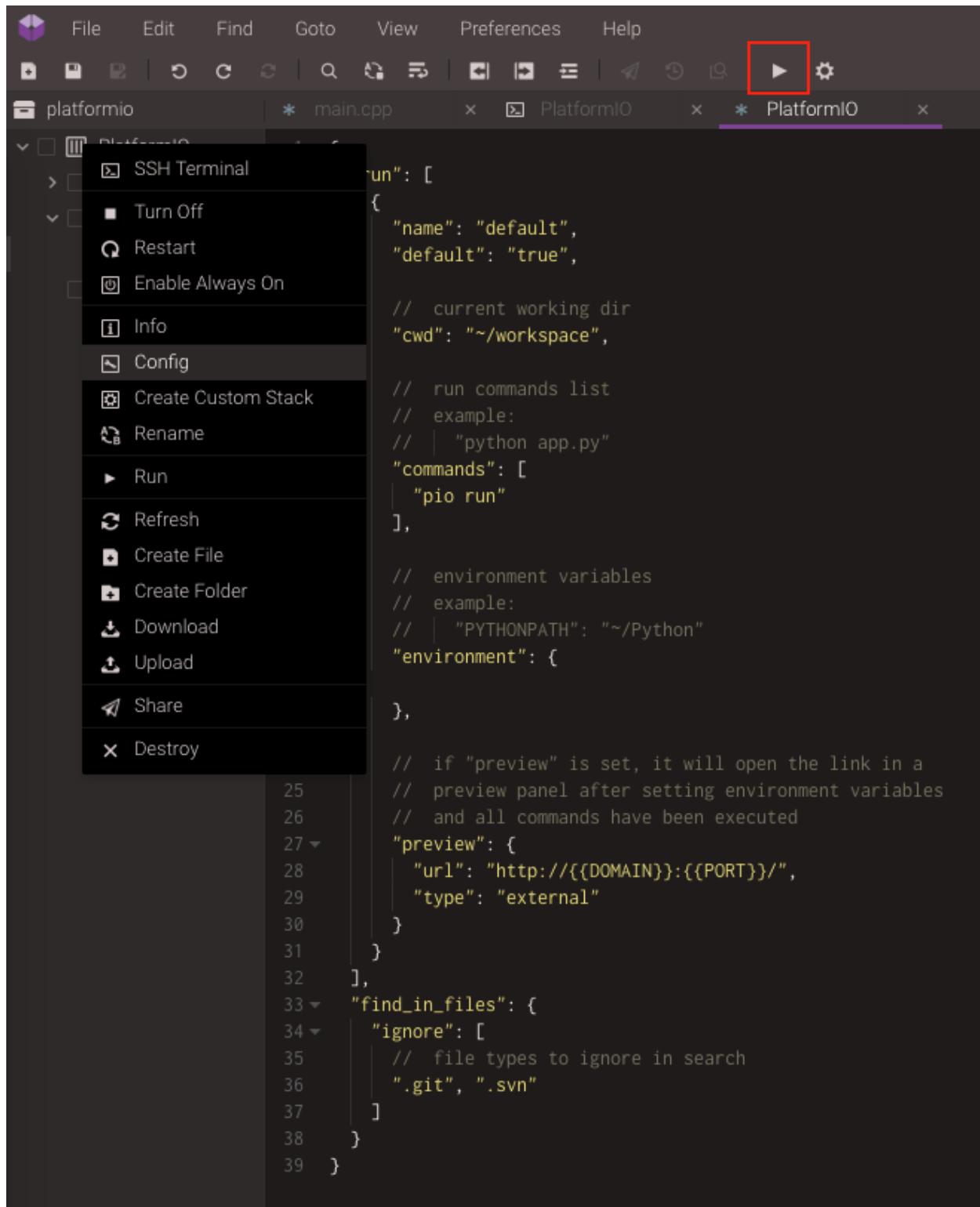
Codeanywhere provides a quick “Run Project” button where you can specify own command. Let’s add “PlatformIO Build Project” command:

1. Open “Project Config” via right click on Container Name (PlatformIO) > Config
2. Set commands field to

```
"commands": [  
    "pio run"  
]
```

3. Save configuration file.

Now, try to click on “Run Project” button. You can assign any PlatformIO command to this button.



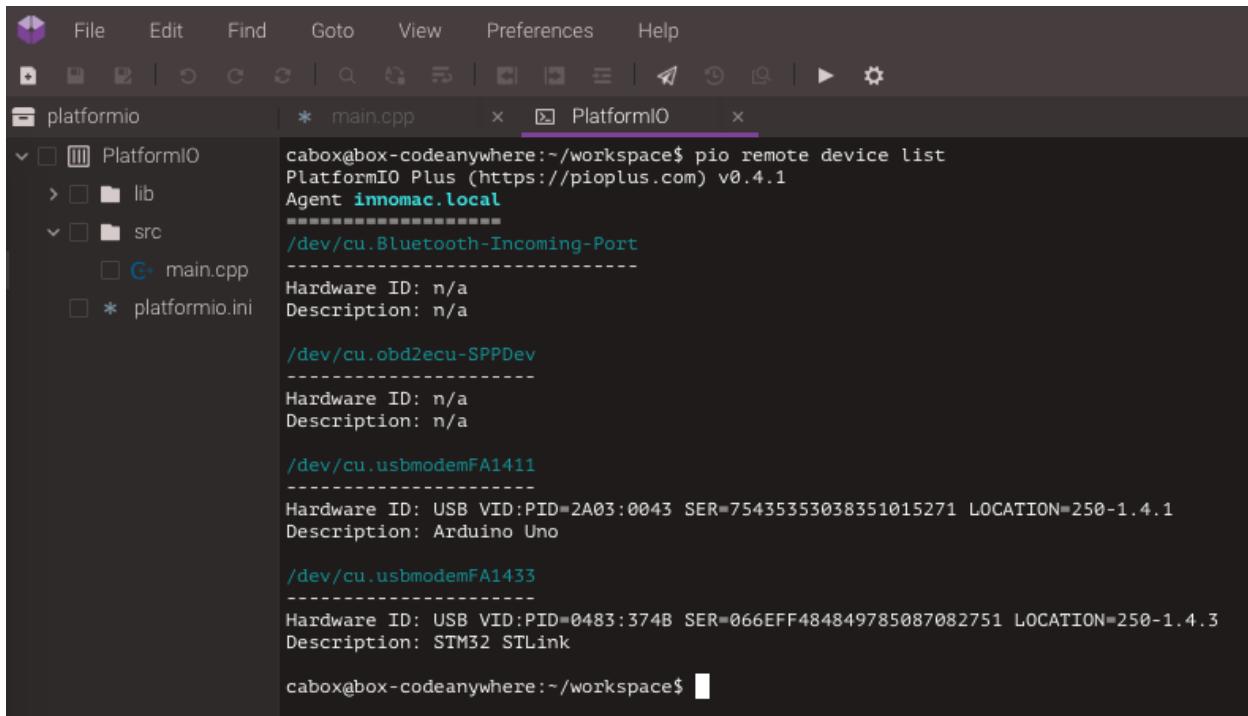
Remote Device Manager

Remote Device Manager works in pair with *PIO Remote*. You can list remote devices that are connected to host machine where *PIO Remote Agent* is started or are visible for it.

1. Open Cloud IDE SSH Terminal

2. Paste this command

```
pio remote device list
```



The screenshot shows the PlatformIO Cloud IDE interface. On the left is a file tree with a project named 'platformio'. The main area displays the output of the 'pio remote device list' command. The output shows a list of connected devices:

```

cabox@box-codeanywhere:~/workspace$ pio remote device list
PlatformIO Plus (https://pioplus.com) v0.4.1
Agent innomac.local
-----
/dev/cu.Bluetooth-Incoming-Port
-----
Hardware ID: n/a
Description: n/a

/dev/cu.obd2ecu-SPPDev
-----
Hardware ID: n/a
Description: n/a

/dev/cu.usbmodemFA1411
-----
Hardware ID: USB VID:PID=2A03:0043 SER=75435353038351015271 LOCATION=250-1.4.1
Description: Arduino Uno

/dev/cu.usbmodemFA1433
-----
Hardware ID: USB VID:PID=0483:374B SER=066EFF484849785087082751 LOCATION=250-1.4.3
Description: STM32 STLink

cabox@box-codeanywhere:~/workspace$
```

Remote Firmware Uploading

Remote Firmware Uploading works in pair with *PIO Remote*. You can deploy firmware to any devices which are visible for *PIO Remote Agent*.

1. Open Cloud IDE SSH Terminal

2. Paste this command

```
pio remote run -t upload
```

```
* main.cpp      x PlatformIO      *
cabox@box-codeanywhere:~/workspace$ pio remote run -t upload
PlatformIO Plus (https://pioplus.com) v0.4.1
Building project locally
[Thu Nov  3 12:04:18 2016] Processing uno (platform: atmelavr, board: uno, framework: arduino)
-----
Verbose mode can be enabled via '-v, --verbose' option
Collected 25 compatible libraries
Looking for dependencies...
Project does not have dependencies
Checking program size
text      data      bss      dec      hex filename
2324        54     168    2546  9f2 .pioenvs/uno/firmware.elf
===== [SUCCESS] Took 0.64 seconds =====
Uploading firmware remotely
[Thu Nov  3 18:04:21 2016] Processing uno (platform: atmelavr, board: uno, framework: arduino)
-----
-----
Verbose mode can be enabled via '-v, --verbose' option
Looking for upload port...
Auto-detected: /dev/cu.usbmodemFA1411
Uploading .pioenvs/uno/firmware.hex

avrduude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.00s

avrduude: Device signature = 0x1e950f
avrduude: reading input file ".pioenvs/uno/firmware.hex"
avrduude: writing flash (2378 bytes):

Writing | ##### | 100% 0.39s

avrduude: 2378 bytes of flash written
avrduude: verifying flash memory against .pioenvs/uno/firmware.hex:
avrduude: load data flash data from input file .pioenvs/uno/firmware.hex:
avrduude: input file .pioenvs/uno/firmware.hex contains 2378 bytes
avrduude: reading on-chip flash data:

Reading | ##### | 100% 0.31s

avrduude: verifying ...
avrduude: 2378 bytes of flash verified

avrduude done. Thank you.
```

Remote Serial Port Monitor

Remote Serial Port Monitor works in pair with [PIO Remote](#). You can read or send data to any device that is connected to host machine where [PIO Remote Agent](#) is started. To list active agents please use this command `platformio remote agent list`.

1. Open Cloud IDE SSH Terminal
2. Paste this command

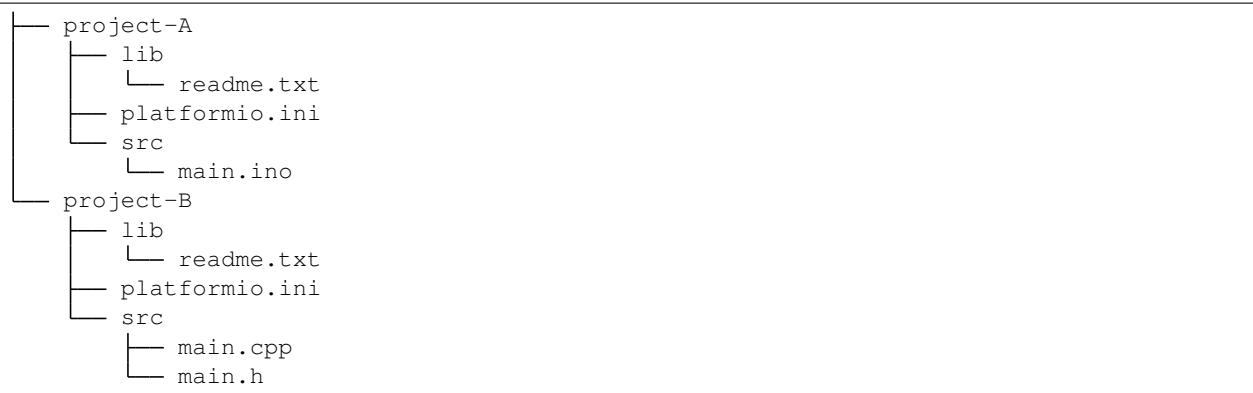
```
pio remote device monitor
```

```
* main.cpp      x PlatformIO      x
cabox@box-codeanywhere:~/workspace$ pio remote device monitor
PlatformIO Plus (https://pioplus.com) v0.4.1
Starting Serial Monitor on innomac.local:/dev/cu.usbmodemFA1411
--- Miniterm on socket://localhost:44128 9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Hello Codeanywhere!
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

--- exit ---
1cabox@box-codeanywhere:~/workspace$
```

Multi-Project workspace

You can have multiple PlatformIO-based Projects in the same workspace. We recommend a next folders structure:



In this case, you need to use `-d`, `--project-dir` option for `platformio run` or `platformio remote run` commands:

- `pio remote run --project-dir project-A -t upload` build Project-A
- **`pio remote run --project-dir project-A -t upload`** remote firmware uploading using Project-A
- **`pio remote run -d project-B -t upload`** remote firmware (program) uploading using Project-B

See documentation for `platformio remote run --project-dir` option.

Eclipse Che

Eclipse Che is an open-source Java based developer workspace server and cloud integrated development environment (IDE) which provides a remote development platform for multi-user purpose. The workspace server comes with a RESTful webservice and provides high flexibility. It also contains a SDK which can be used to create plug-ins for languages, frameworks or tools.

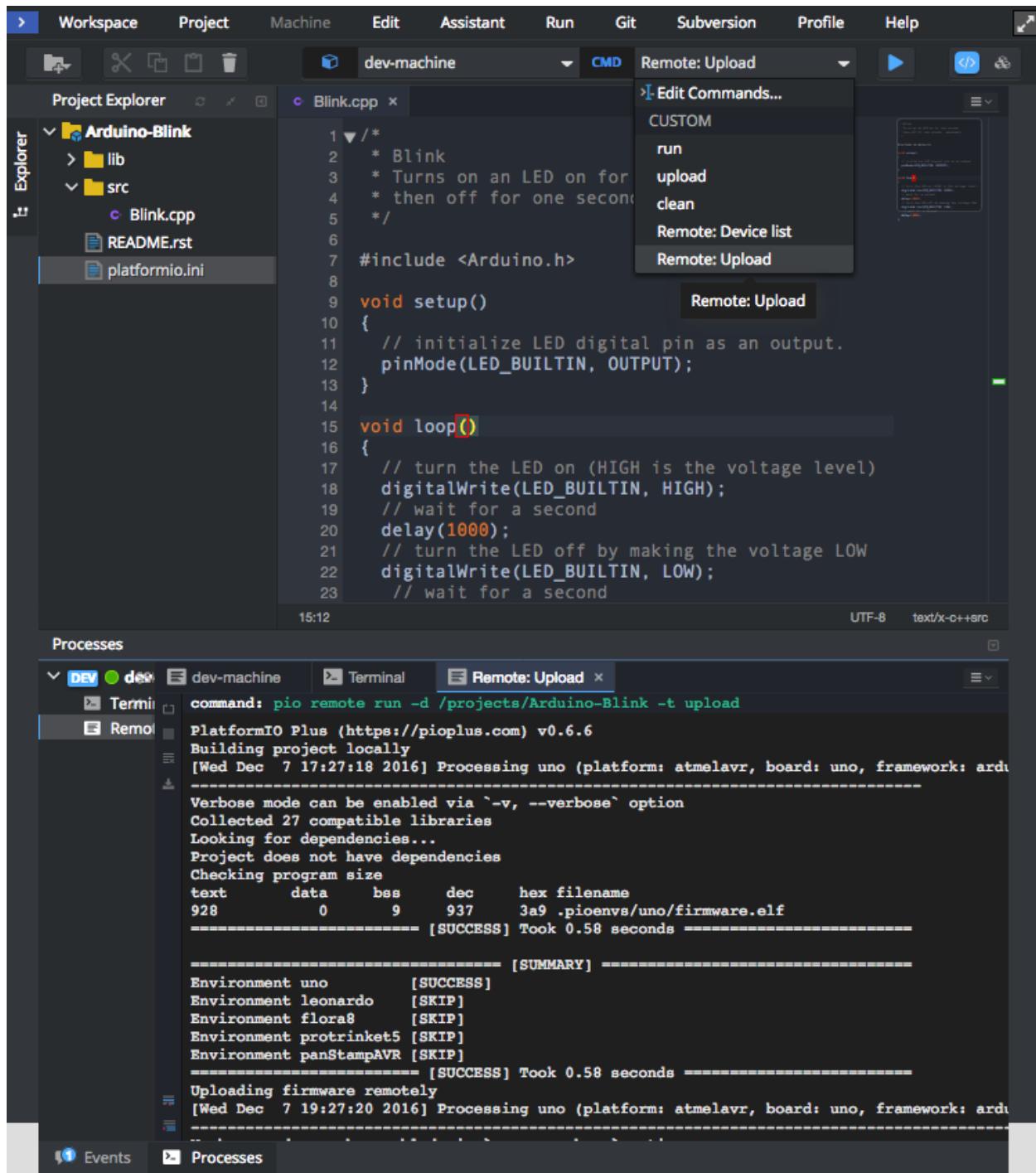
Contents

- *Eclipse Che*
 - *Demo*
 - *Integration*
 - *Quick Start*
 - *Multi-Project workspace*

Note:

1. Please make sure to read [PIO Remote](#) guide first.
 2. You need [PIO Account](#) if you don't have it. Registration is FREE.
 3. You should have a running [PIO Remote Agent](#) on a remote machine where hardware devices are connected physically or accessible for the remote operations. See [PIO Remote Quick Start](#) for details.
-

Demo



Integration

1. Sign in to Codenvy (based on Eclipse Che). A registration is FREE and gives you unlimited private projects.
2. Open Workspaces tab

3. Click on “Add Workspace”, then switch to “Runtime” tab.
 - **Name** set to “PlatformIO”
 - **Stack** search for PLATFORMIO
 - Click on “Create” button, then “Open”.

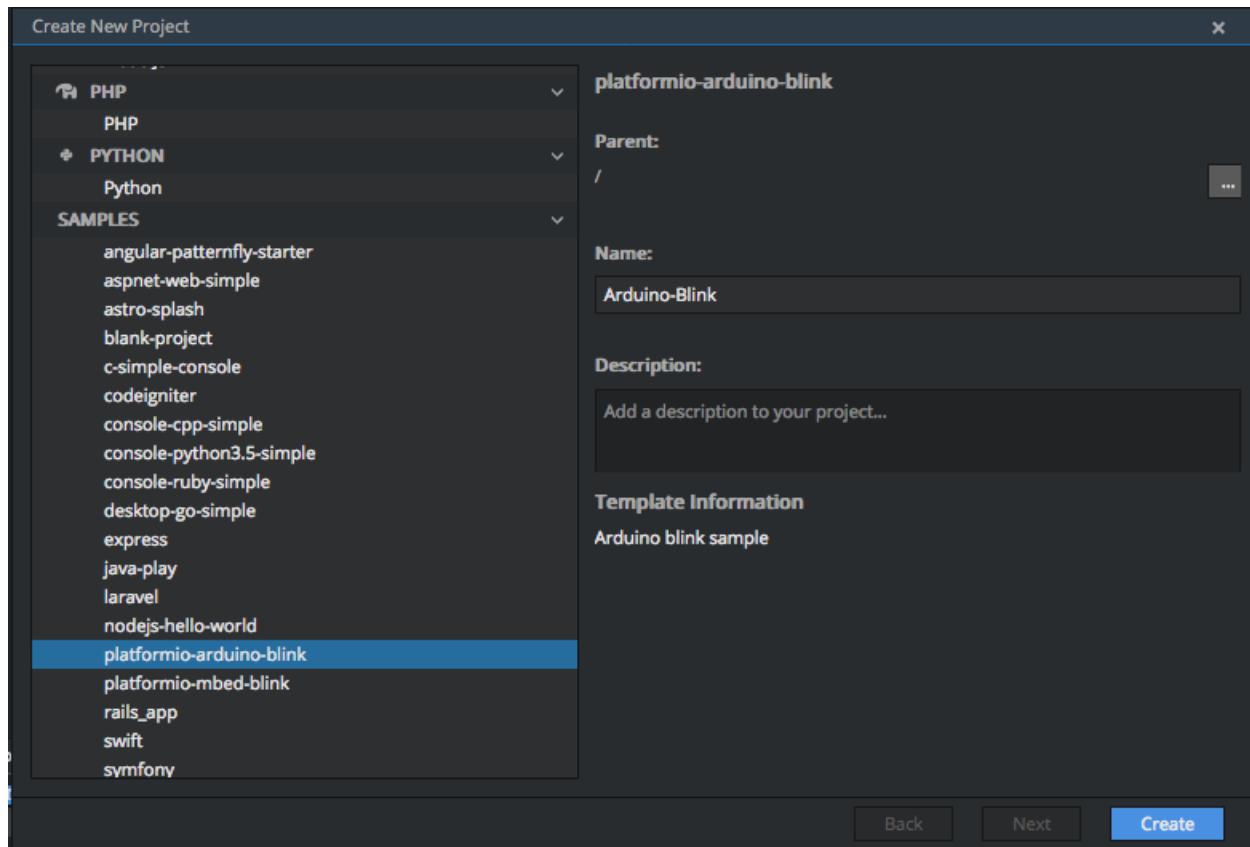
The screenshot shows the Codenvy interface. On the left, there's a sidebar with 'Dashboard', 'Workspaces' (selected), 'Stacks', and 'Factories'. The main area is titled 'New Workspace' with a 'CREATE' button. Below it, there are 'Settings' and 'Runtime' tabs, with 'Runtime' selected. A 'Name' field contains 'PlatformIO'. Under 'Select Stack', there's a description about stacks being recipes or images used to define your environment runtime. A 'Filter' section shows 'You can filter the stacks by selecting technologies.' with 'PLATFORMIO' selected. A 'Ready-to-go Stacks' section lists 'Stack 1' (PLATFORMIO). A detailed view of the 'PLATFORMIO' stack is shown in a box, featuring its logo, name, and description: 'PlatformIO, IoT, embedded, arduino, mbed'.

4. Using opened Terminal, please log in to *PIO Account* using *platformio account login* command.

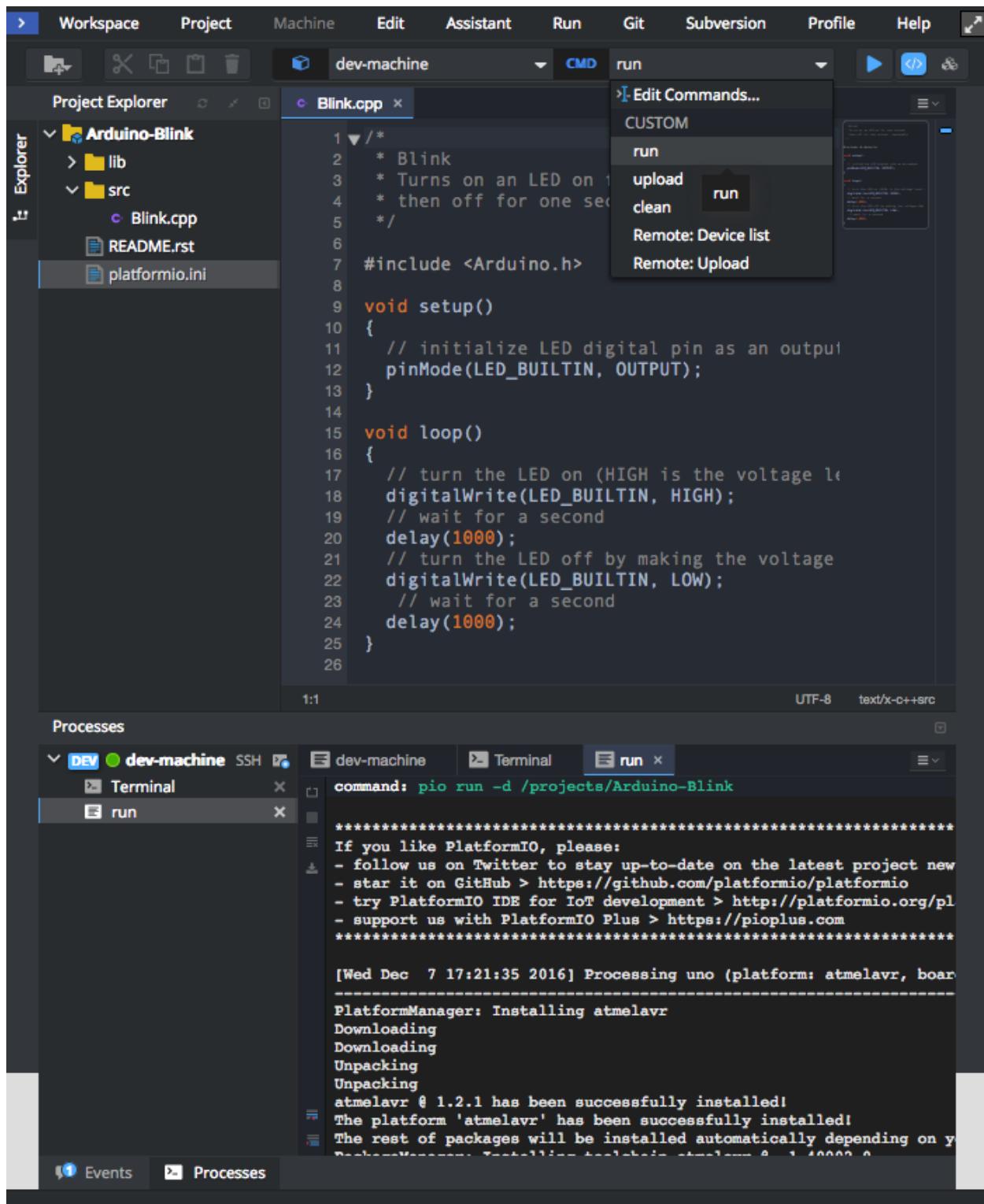
Quick Start

Let's create our first PlatformIO-based Codenvy Project

1. Click on Menu: Workspace > Create New Project and select `platformio-arduino-blink` sample. Set “Name” to “Arduino Blink” and press “Create”.



2. Now you can use dropdown Commands menu and process project with “run” command



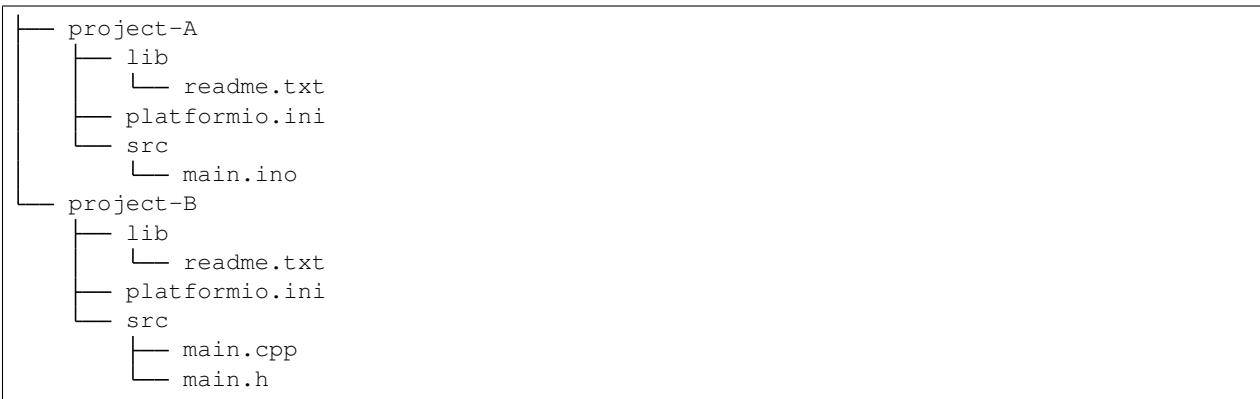
3. If you prefer to work with *PlatformIO Core* CLI, then you can process project using Cloud IDE Terminal and the next commands:

- *platformio run* - build project locally (using Cloud IDE's virtual machine)
- *pio run -t clean* - clean project

- `pio remote run -t upload` - upload firmware (program) to a remote device
- `platformio remote device list` - list available remote devices
- `platformio remote device monitor` - Remote Serial Port Monitor

Multi-Project workspace

You can have multiple PlatformIO-based Projects in the same workspace. We recommend a next folders structure:



In this case, you need to use `-d`, `--project-dir` option for `platformio run` or `platformio remote run` commands:

- `pio remote run --project-dir project-A -t upload` build Project-A
- `pio remote run --project-dir project-A -t upload` remote firmware uploading using Project-A
- `pio remote run -d project-B -t upload` remote firmware (program) uploading using Project-B

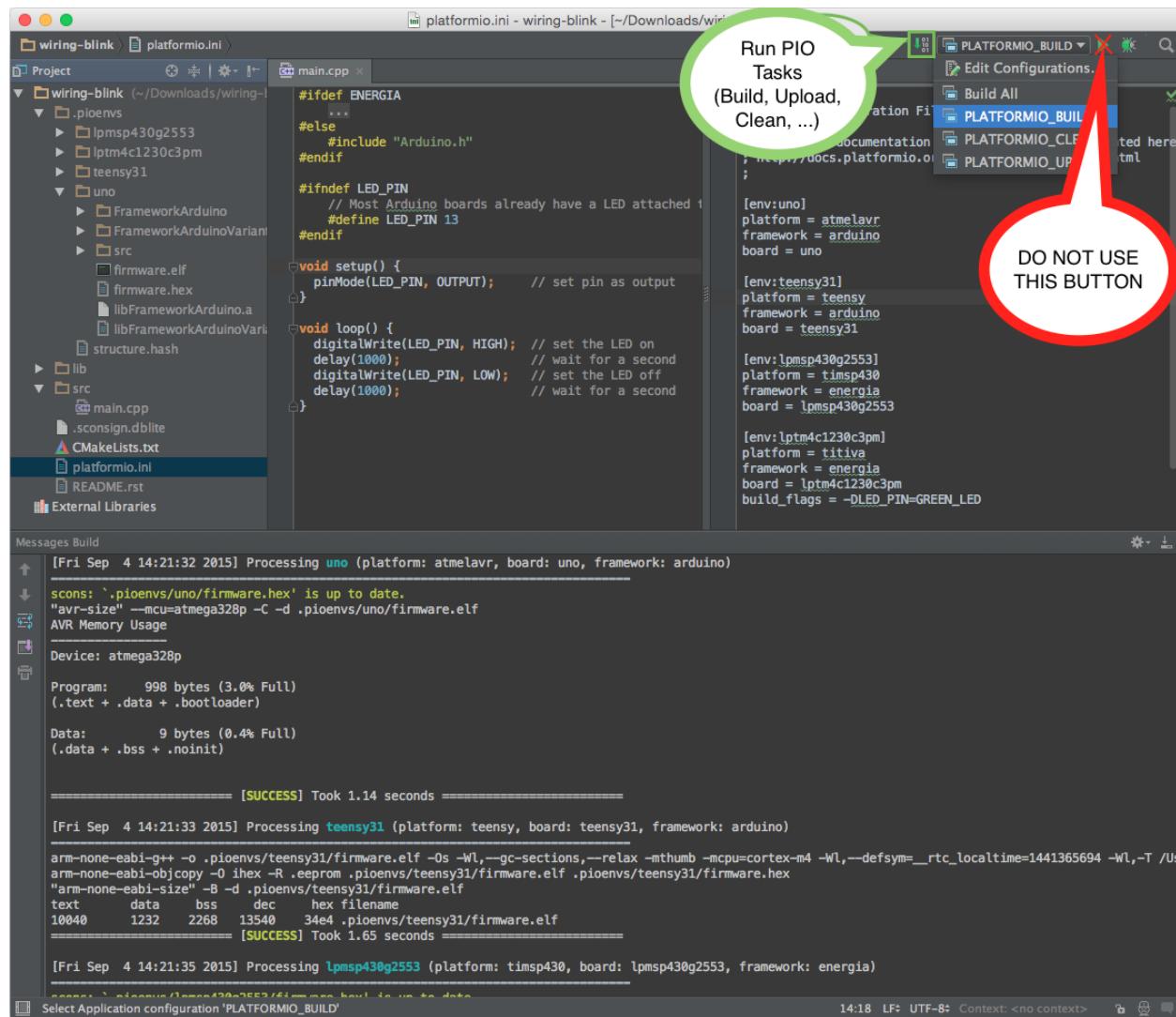
See documentation for `platformio remote run --project-dir` option.

1.18.3 Desktop IDE

CLion

The CLion is a cross-platform C/C++ IDE for Linux, OS X, and Windows integrated with the CMake build system. The initial version will support the GCC and Clang compilers and GDB debugger. Clion includes such features as a smart editor, code quality assurance, automated refactorings, project manager, integrated version control systems.

Refer to the [CLion Documentation](#) page for more detailed information.



Contents

- *Integration*
- *Known issues*
 - Arduino .ino files are not supported
- *Articles / Manuals*

Integration

Integration process consists of these steps:

1. Open system Terminal and install *PlatformIO Core*
2. Create new folder for your project and change directory (`cd`) to it
3. Generate a project using PIO Core Project Generator (`platformio init --ide`)

4. Import project in IDE.

Choose board ID using *platformio boards* or Embedded Boards Explorer command and generate project via *platformio init --ide* command:

```
platformio init --ide clion --board <ID>
# For example, generate project for Arduino Uno
platformio init --ide clion --board uno
```

Then:

1. Place source files (*.c, *.cpp, *.h, *.hpp) to `src` directory and repeat *platformio init --ide* command above (to refresh source files list)
2. Import this project via Menu: File > Import Project and specify root directory where is located *Project Configuration File platformio.ini*
3. Open source file from `src` directory
4. Build project (*DO NOT* use “Run” button, see marks on the screenshot above): Menu: Run > Build.

Warning:

1. *PlatformIO Core DOES NOT* depend on Cmake, it has own cross-platform Build System. All data related to build flags and source code filtering should be specified using *Build options* in *Project Configuration File platformio.ini*.
2. See know issue: *Arduino .ino files are not supported* and how to resolve it.

There are 6 predefined targets for building (*NOT FOR RUNNING*, see marks on the screenshot above):

- PLATFORMIO_BUILD - Build project without auto-uploading
- PLATFORMIO_UPLOAD - Build and upload (if no errors)
- PLATFORMIO_CLEAN - Clean compiled objects
- PLATFORMIO_TEST - *PIO Unit Testing*
- PLATFORMIO_PROGRAM - Build and upload using external programmer (if no errors), see *Upload using Programmer*
- PLATFORMIO_UPLOADFS - Upload files to file system SPIFFS, see *Uploading files to file system SPIFFS*
- PLATFORMIO_UPDATE - Update installed platforms and libraries via *platformio update*
- PLATFORMIO_REBUILD_PROJECT_INDEX - Rebuild C/C++ Index for the Project. Allows to fix code completion and code linting issues.

PlatformIO Project Generator created “File Watchers” configuration to monitor changes in `platformio.ini` and automatically rebuild C/C++ Project Index. **You need to install extra plugin** named File Watchers via “Clion: Preferences > Plugins” to enable this feature.

Warning: The libraries which are added, installed or used in the project after generating process will not be reflected in IDE. To fix it please run `PLATFORMIO_REBUILD_PROJECT_INDEX` target.

Known issues

Arduino .ino files are not supported

CLion uses “CMake” tool for code completion and code linting. As result, it doesn’t support Arduino files (*.ino and .pde) because they are not valid C/C++ based source files:

1. Missing includes such as `#include <Arduino.h>`
2. Function declarations are omitted.

See how to [Convert Arduino file to C++ manually](#).

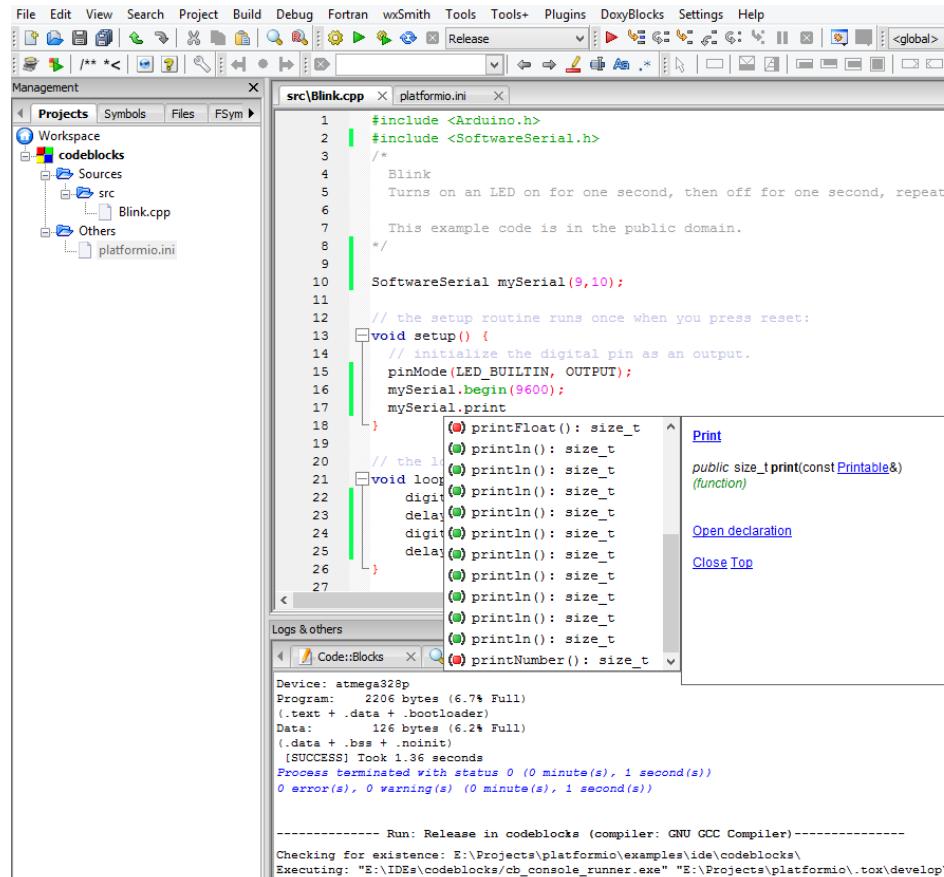
Articles / Manuals

- Dec 01, 2015 - **JetBrains CLion Blog** - C++ Annotated: Fall 2015. Arduino Support in CLion using PlatformIO
- Nov 22, 2015 - **Michał Seroczyński** - Using PlatformIO to get started with Arduino in CLion IDE
- Nov 09, 2015 - **ÁLvaro García Gómez** - Programar con Arduino “The good way” (Programming with Arduino “The good way”, Spanish)

See more [Articles about us](#).

CodeBlocks

Code::Blocks is a free, open-source cross-platform IDE that supports multiple compilers including GCC, Clang and Visual C++. It is developed in C++ using wxWidgets as the GUI toolkit. Using a plugin architecture, its capabilities and features are defined by the provided plugins. Currently, Code::Blocks is oriented towards C, C++, and Fortran.



CodeBlocks IDE can be downloaded from [here](#).

Contents

- *CodeBlocks*
 - *Integration*

Integration

Integration process consists of these steps:

1. Open system Terminal and install *PlatformIO Core*
2. Create new folder for your project and change directory (`cd`) to it
3. Generate a project using PIO Core Project Generator (`platformio init --ide`)
4. Import project in IDE.

Choose board ID using `platformio boards` or Embedded Boards Explorer command and generate project via `platformio init --ide` command:

```
platformio init --ide codeblocks --board <ID>
```

(continues on next page)

(continued from previous page)

```
# For example, generate project for Arduino UNO
platformio init --ide codeblocks --board uno
```

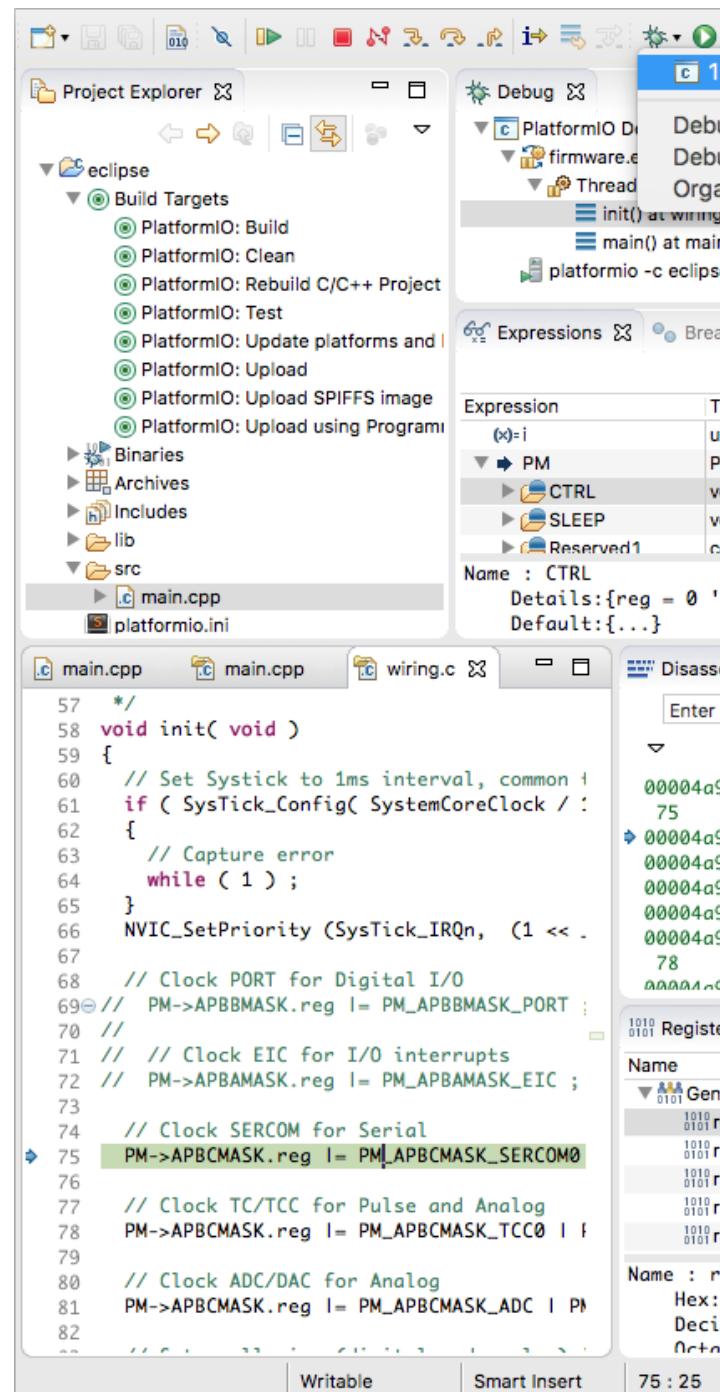
Then:

1. Open this project via Menu: File > Open...
2. Add new files to `src` directory (`*.c`, `*.cpp`, `*.ino`, etc.) via Menu: File > New > File..
3. Build project using Menu: Build > Build
4. Upload firmware using Menu: Build > Run

Warning: The libraries which are added, installed or used in the project after generating process wont be reflected in IDE. To fix it you need to reinitialize project using `platformio init` (repeat it).

Eclipse

The [Eclipse CDT \(C/C++ Development Tooling\)](#) Project provides a fully functional C and C++ Integrated Development Environment based on the Eclipse platform. Features include: support for project creation and managed build for various toolchains, standard make build, source navigation, various source knowledge tools, such as type hierarchy, call graph, include browser, macro definition browser, code editor with syntax highlighting, folding and hyperlink navigation, source code refactoring and code generation, visual debugging tools, including memory, registers, and disassembly viewers.



Refer to the [CDT Documentation](#) page for more detailed information.

Contents

- [Integration](#)
- [Live Integration](#)
- [Debugging](#)
- [Articles / Manuals](#)

Integration

Integration process consists of these steps:

1. Open system Terminal and install [PlatformIO Core](#)
 2. Create new folder for your project and change directory (`cd`) to it
 3. Generate a project using PIO Core Project Generator (`platformio init --ide`)
 4. Import project in IDE.
-

Choose board ID using [platformio boards](#) or Embedded Boards Explorer command and generate project via `platformio init --ide` command:

```
platformio init --ide eclipse --board <ID>

# For example, generate project for Arduino Uno
platformio init --ide eclipse --board uno
```

Then:

1. Import this project via Menu: File > Import... > General > Existing Projects into Workspace > Next and specify root directory where is located [Project Configuration File platformio.ini](#)
2. Open source file from `src` directory (`*.c`, `*.cpp`, `*.ino`, etc.)
3. Build project using Menu: Project > Build Project or pre-configured Make Targets (see screenshot below):
 - PlatformIO: Build - Build project without auto-uploading
 - PlatformIO: Clean - Clean compiled objects.
 - PlatformIO: Test - [PIO Unit Testing](#)
 - PlatformIO: Upload - Build and upload (if no errors)
 - PlatformIO: Upload using Programmer see [Upload using Programmer](#)
 - PlatformIO: Upload SPIFFS image see [Uploading files to file system SPIFFS](#)
 - PlatformIO: Update platforms and libraries - Update installed platforms and libraries via [platformio update](#)
 - PlatformIO: Rebuild C/C++ Project Index - Rebuild C/C++ Index for the Project. Allows to fix code completion and code linting issues.

If you have some problems with unresolved includes, defines, etc., then

1. Rebuild PlatformIO Project Index: PlatformIO: Rebuild C/C++ Project Index target
2. Rebuild Eclipse Project Index: Menu: Project > C/C++ Index > Rebuild
3. Refresh Project, right click on the project Project > Refresh (F5) or restart Eclipse IDE.

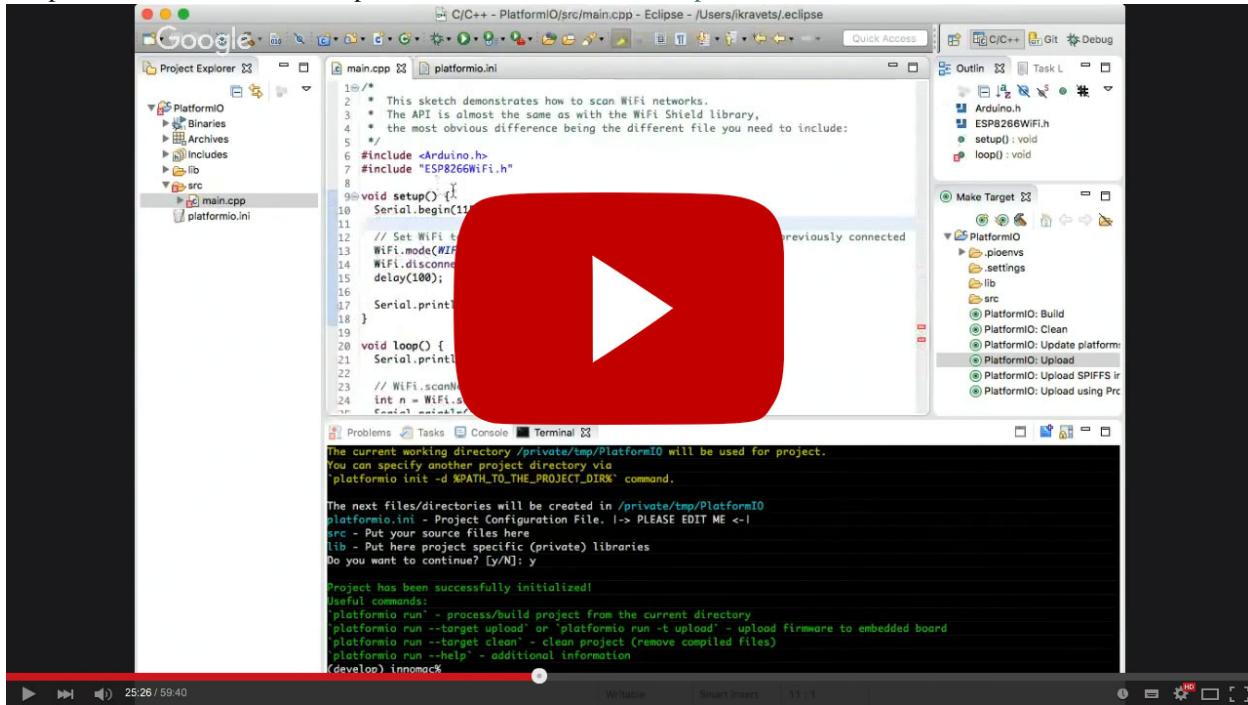
Warning: The libraries which are added, installed or used in the project after generating process wont be reflected in IDE. To fix it please run PlatformIO: Rebuild C/C++ Project Index target and right click on the project and Project > Refresh (F5).

Warning: The C/C++ GCC Cross Compiler Support package must be installed in Eclipse, otherwise the CDT Cross GCC Built-in Compiler Settings provider will not be available (check the Providers tab in Project > Properties > C/C++ General > Preprocessor Include Paths, Macros etc. for a marked entry named CDT Cross GCC Built-in Compiler Settings).

If this provider is not available, toolchain related includes cannot be resolved.

Live Integration

Eclipse Virtual IoT Meetup: PlatformIO: a cross-platform IoT solution to build them all!



Debugging

A debugging feature is provided by [PIO Unified Debugger](#) and new debug configuration named “PlatformIO Debugger” is created. No need to do extra configuration steps!

1. Build a project first time or after “Clean” operation using PlatformIO: Build target
2. Launch debugger via “Menu: Debug” or “Bug Icon” button on Tool Bar.
3. Wait for a while, PlatformIO will prepare project for debugging and session will be started soon.

Articles / Manuals

- May 05, 2016 - **Ivan Kravets, Ph.D. / Eclipse Virtual IoT Meetup** - PlatformIO: a cross-platform IoT solution to build them all!
- Sep 01, 2015 - **Thomas P. Weldon, Ph.D.** - [Improvised MBED FRDM-K64F Eclipse/PlatformIO Setup and Software Installation](#)

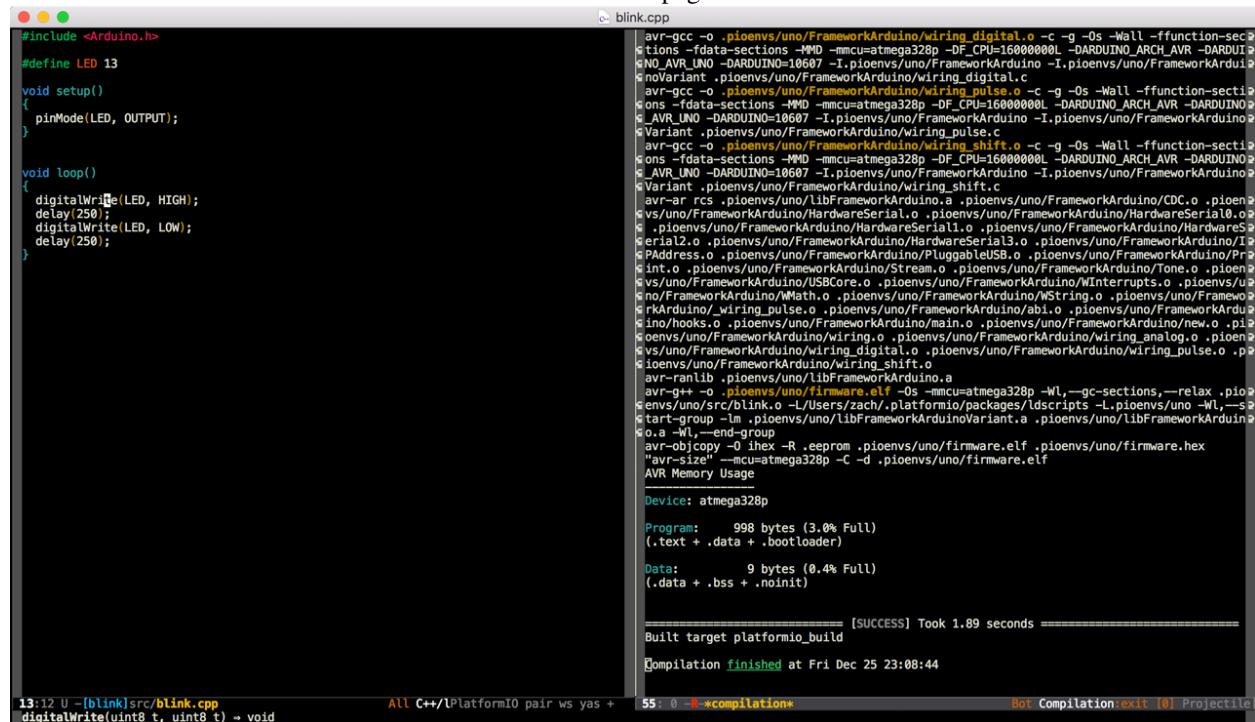
- Jul 11, 2015 - **TrojanC** - Learning Arduino GitHub Repository
- June 20, 2014 - **Ivan Kravets, Ph.D.** - Building and debugging Atmel AVR (Arduino-based) project using Eclipse IDE+PlatformIO

See a full list with [Articles about us](#).

Emacs

GNU Emacs is an extensible, customizable text editor - and more. At its core is an interpreter for Emacs Lisp, a dialect of the [Lisp programming language](#) with extensions to support text editing.

Refer to the Emacs Documentation page for more detailed information.



```
#include <Arduino.h>
#define LED 13
void setup()
{
  pinMode(LED, OUTPUT);
}
void loop()
{
  digitalWrite(LED, HIGH);
  delay(250);
  digitalWrite(LED, LOW);
  delay(250);
}
```

```
avr-gcc -o .pioenvs/uno/FrameworkArduino/wiring_digital.o -c -g -Os -Wall -ffunction-sections -fdata-sections -MM -mcpu=atmega328p -DF_CPU=16000000L -DARDUINO_ARCH_AVR -DARDUINO=10607 -I.pioenvs/uno/FrameworkArduino -I.pioenvs/uno/FrameworkArduinoVariant.pioenvs/uno/FrameworkArduino/wiring_digital.c
avr-gcc -o .pioenvs/uno/FrameworkArduino/wiring_pulse.o -c -g -Os -Wall -ffunction-sections -fdata-sections -MM -mcpu=atmega328p -DF_CPU=16000000L -DARDUINO_ARCH_AVR -DARDUINO=10607 -I.pioenvs/uno/FrameworkArduino -I.pioenvs/uno/FrameworkArduinoVariant.pioenvs/uno/FrameworkArduino/wiring_pulse.c
avr-gcc -o .pioenvs/uno/FrameworkArduino/wiring_shift.o -c -g -Os -Wall -ffunction-sections -fdata-sections -MM -mcpu=atmega328p -DF_CPU=16000000L -DARDUINO_ARCH_AVR -DARDUINO=10607 -I.pioenvs/uno/FrameworkArduino -I.pioenvs/uno/FrameworkArduinoVariant.pioenvs/uno/FrameworkArduino/wiring_shift.c
avr-ar rcs .pioenvs/uno/libFrameworkArduino.a .pioenvs/uno/FrameworkArduino/CDC.o .pioenvs/uno/FrameworkArduino/HardwareSerial.o .pioenvs/uno/FrameworkArduino/HardwareSerial1.o .pioenvs/uno/FrameworkArduino/HardwareSerial2.o .pioenvs/uno/FrameworkArduino/HardwareSerial3.o .pioenvs/uno/FrameworkArduino/13Address.o .pioenvs/uno/FrameworkArduino/PluggableUSB.o .pioenvs/uno/FrameworkArduino/PortInt.o .pioenvs/uno/FrameworkArduino/Wiring.o .pioenvs/uno/FrameworkArduino/WiringAnalog.o .pioenvs/uno/FrameworkArduino/WiringDigital.o .pioenvs/uno/FrameworkArduino/WiringPulse.o .pioenvs/uno/FrameworkArduino/WiringShift.o
avr-ranlib .pioenvs/uno/libFrameworkArduino.a
avr-gcc -o .pioenvs/uno/firmware.elf -Os -mcpu=atmega328p -Wl,-gc-sections,--relax .pioenvs/uno/src/blink.o -L/Users/zach/.platformio/packages/ldscripts -L.pioenvs/uno -Wl,--start-group -lpioenvs/uno/libFrameworkVariant.a .pioenvs/uno/libFrameworkArduino.a -Wl,--end-group
avr-objcopy -O ihex -R .eeprom .pioenvs/uno/firmware.elf .pioenvs/uno/firmware.hex
"AVR-size" --mcu=atmega328p -C -d .pioenvs/uno/firmware.elf
AVR Memory Usage
Device: atmega328p
Program:    998 bytes (3.0% Full)
(.text + .data + .bootloader)
Data:        9 bytes (0.4% Full)
(.data + .bss + .noinit)

=====
[BUILT] [SUCCESS] Took 1.89 seconds
Built target platformio_build
Compilation finished at Fri Dec 25 23:08:44
55: 0 -R-*compilation* Bot Compilation:exit [0] Projectile
```

Contents

- *Emacs*
 - *Integration*
 - * *PlatformIO-Mode*
 - * *Project Generator*

Integration

Integration process consists of these steps:

1. Open system Terminal and install [PlatformIO Core](#)
2. Create new folder for your project and change directory (`cd`) to it

3. Generate a project using PIO Core Project Generator (`platformio init --ide`)
 4. Import project in IDE.
-

PlatformIO-Mode

An Emacs minor mode has been written to facilitate building and uploading from within Emacs. It can be installed from the MELPA repository using `M-x package-install`. See the MELPA [Getting Started](#) page for more information.

Setup instructions and an example config can be found at the [Github](#) page.

Code completion can optionally be provided by installing `irony-mode`

Project Generator

Choose board ID using `platformio boards` or Embedded Boards Explorer command and generate project via `platformio init --ide` command:

```
platformio init --ide emacs --board <ID>
```

There are 6 predefined targets for building.

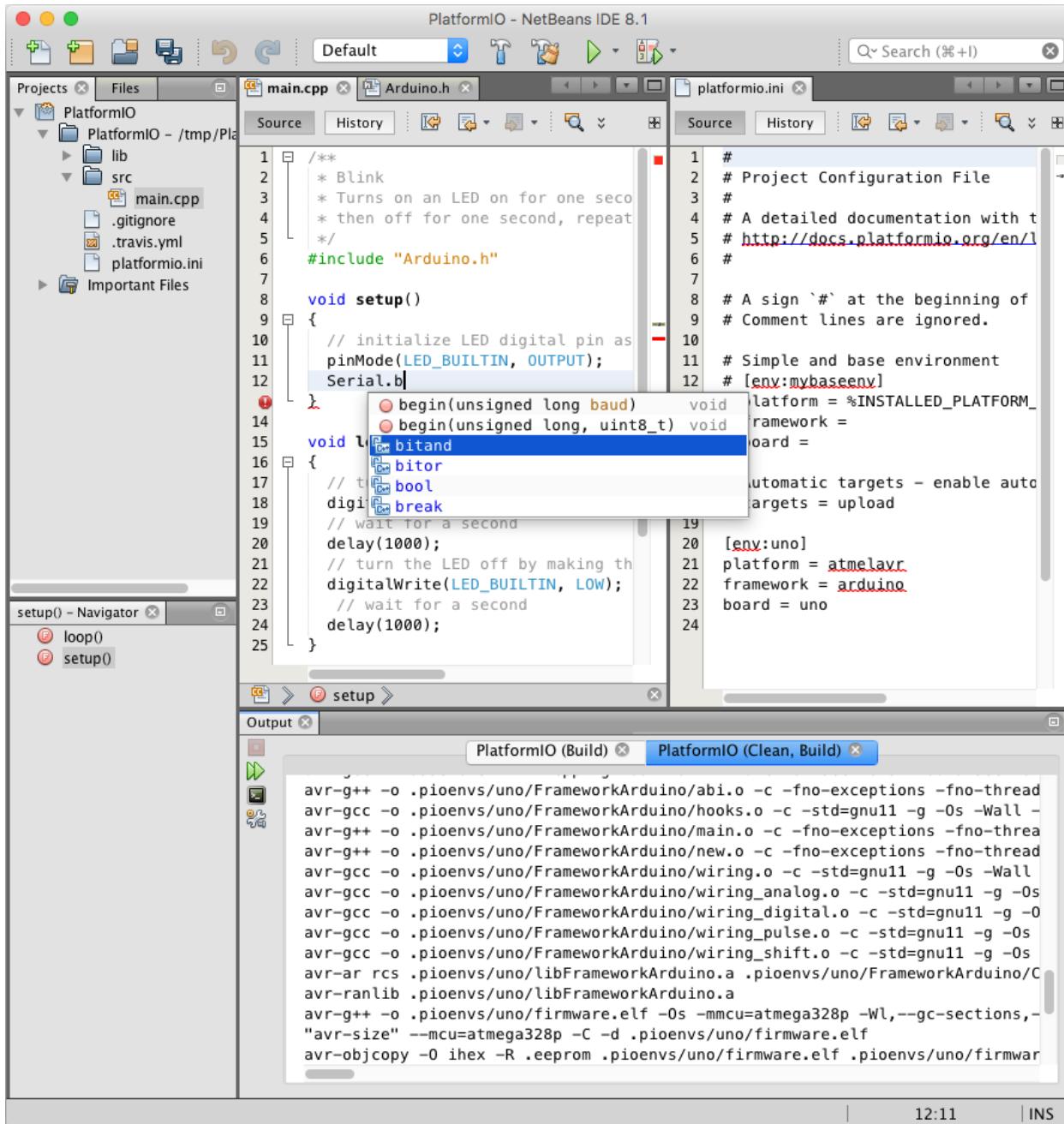
- `platformio_build` - Build project without auto-uploading. (`C-c i b`)
- `platformio_clean` - Clean compiled objects. (`C-c i c`)
- `platformio_upload` - Build and upload (if no errors). (`C-c i u`)
- `platformio_programmer_upload` - Build and upload using external programmer (if no errors, see [Upload using Programmer](#)). (`C-c i p`)
- `platformio_spiffs_upload` - Upload files to file system SPIFFS (see [Uploading files to file system SPIFFS](#)). (`C-c i s`)
- `platformio_update` - Update installed platforms and libraries. (`C-c i d`)

Warning: The libraries which are added, installed or used in the project after generating process wont be reflected in IDE. To fix it you need to reinitialize project using `platformio init` (repeat it).

NetBeans

NetBeans is a Java-based integrated development environment (IDE). It provides out-of-the-box code analyzers and editors for working with the latest Java 8 technologies—Java SE 8, Java SE Embedded 8, and Java ME Embedded 8. The IDE also has a range of new tools for HTML5/JavaScript, in particular for Node.js, KnockoutJS, and AngularJS; enhancements that further improve its support for Maven and Java EE with PrimeFaces; and improvements to PHP and C/C++ support.

NetBeans IDE can be downloaded from [here](#). Just make sure you download the C/C++ version (or if you already use NetBeans, install the C/C++ development plugins).



Contents

- *NetBeans*
 - *Integration*
 - *Articles / Manuals*

Integration

Integration process consists of these steps:

1. Open system Terminal and install *PlatformIO Core*
2. Create new folder for your project and change directory (`cd`) to it
3. Generate a project using PIO Core Project Generator (`platformio init --ide`)
4. Import project in IDE.

Choose board ID using *platformio boards* or Embedded Boards Explorer command and generate project via `platformio init --ide` command:

```
platformio init --ide netbeans --board <ID>
# For example, generate project for Arduino Uno
platformio init --ide netbeans --board uno
```

Then:

1. Open this project via Menu: File > Open Project...
2. Add new files to `src` directory (`*.c`, `*.cpp`, `*.ino`, etc.) via right-click on `src` folder in the “Projects” pane
3. Build project using Menu: Run > Build Project
4. Upload firmware using Menu: Run > Run Project

Warning: The libraries which are added, installed or used in the project after generating process wont be reflected in IDE. To fix it you need to reinitialize project using `platformio init` (repeat it).

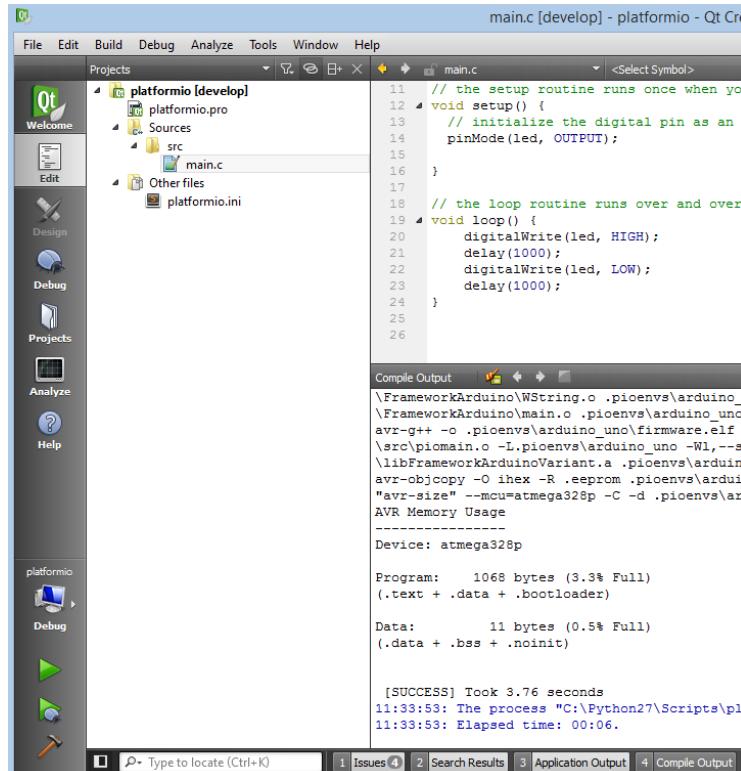
Articles / Manuals

- Feb 22, 2016 - **Grzegorz Holdys** - How to Integrate PlatformIO with Netbeans

See the full list with [Articles about us](#).

Qt Creator

The [Qt Creator](#) is an open source cross-platform integrated development environment. The editor includes such features as syntax highlighting for various languages, project manager, integrated version control systems, rapid code navigation tools and code autocompletion.



Refer to the [Qt-creator Manual](#) page for more detailed information.

Contents

- *Qt Creator*
 - *Integration*
 - * *Project Generator*
 - * *Manual Integration*
 - *Setup New Project*
 - *First program in Qt Creator*
 - *Conclusion*

Integration

Integration process consists of these steps:

1. Open system Terminal and install *PlatformIO Core*
 2. Create new folder for your project and change directory (`cd`) to it
 3. Generate a project using PIO Core Project Generator (`platformio init --ide`)
 4. Import project in IDE.

Project Generator

Choose board ID using `platformio boards` or Embedded Boards Explorer command and generate project via `platformio init --ide` command:

```
platformio init --ide qtcreator --board <ID>

# For example, generate project for Arduino Uno
platformio init --ide qtcreator --board uno
```

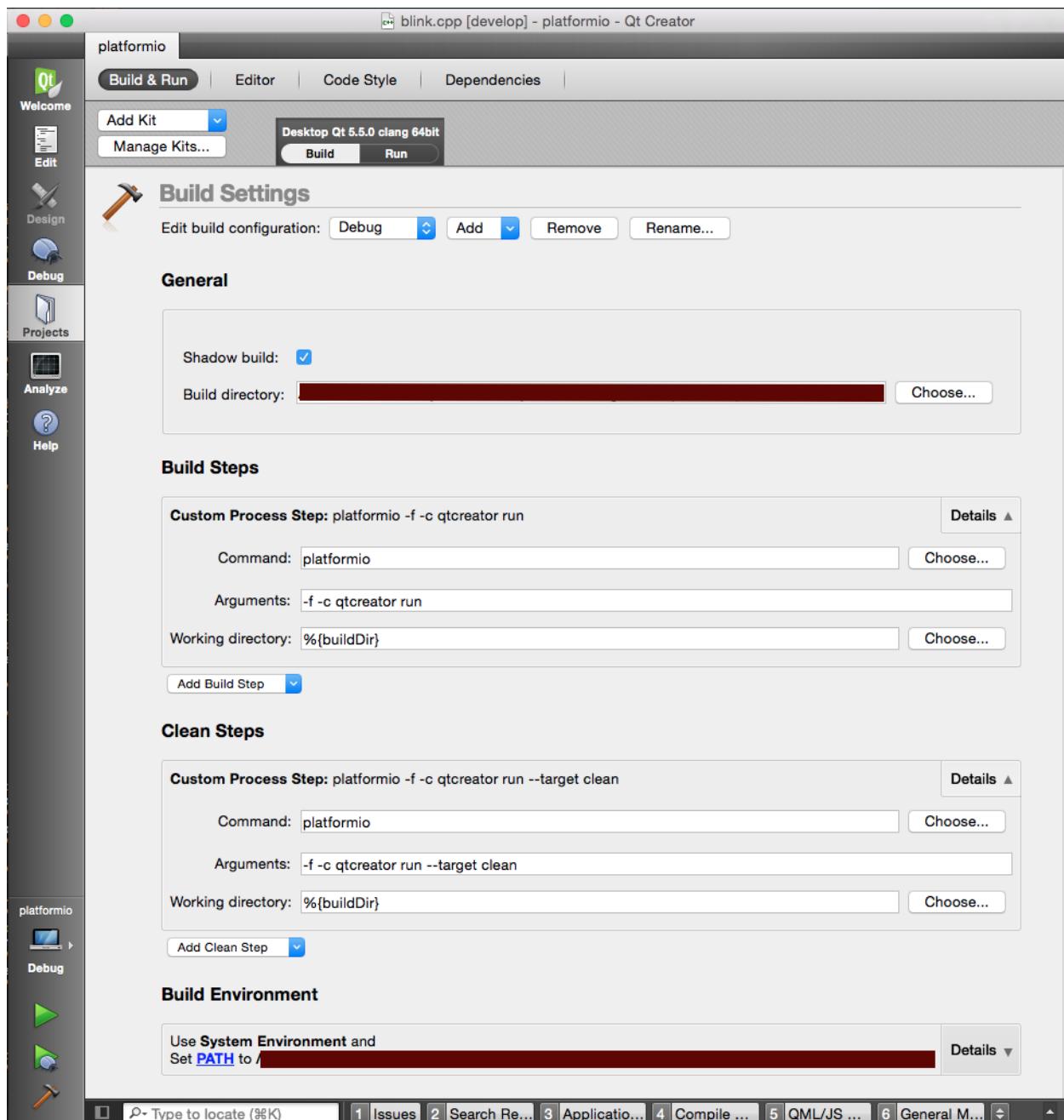
Then:

1. Import project via File > Open File or Project and select `platformio.pro` from the folder where is located *Project Configuration File `platformio.ini`*
2. Select default desktop kit and click on Configure Project (Projects mode, left panel)
3. Set General > Build directory to the project directory where is located *Project Configuration File `platformio.ini`*
4. Remove all items from Build Steps, click on Build Steps > Add Build Step > Custom Process Step and set:
 - **Command:** `platformio`
 - **Arguments:** `-f -c qtcreator run`
 - **Working directory:** `%{buildDir}`
5. Remove all items from Clean Steps, click on Clean Steps > Add Clean Step > Custom Process Step and set:
 - **Command:** `platformio`
 - **Arguments:** `-f -c qtcreator run --target clean`
 - **Working directory:** `%{buildDir}`
6. Update PATH in Build Environment > PATH > EDIT with the result of this command (paste in Terminal):

```
# Linux, Mac
echo $PATH

# Windows
echo %PATH%
```

7. Switch to Edit mode (left panel) and open source file from `src` directory (`*.c`, `*.cpp`, `*.ino`, etc.)
8. Build project: Menu: Build > Build All.

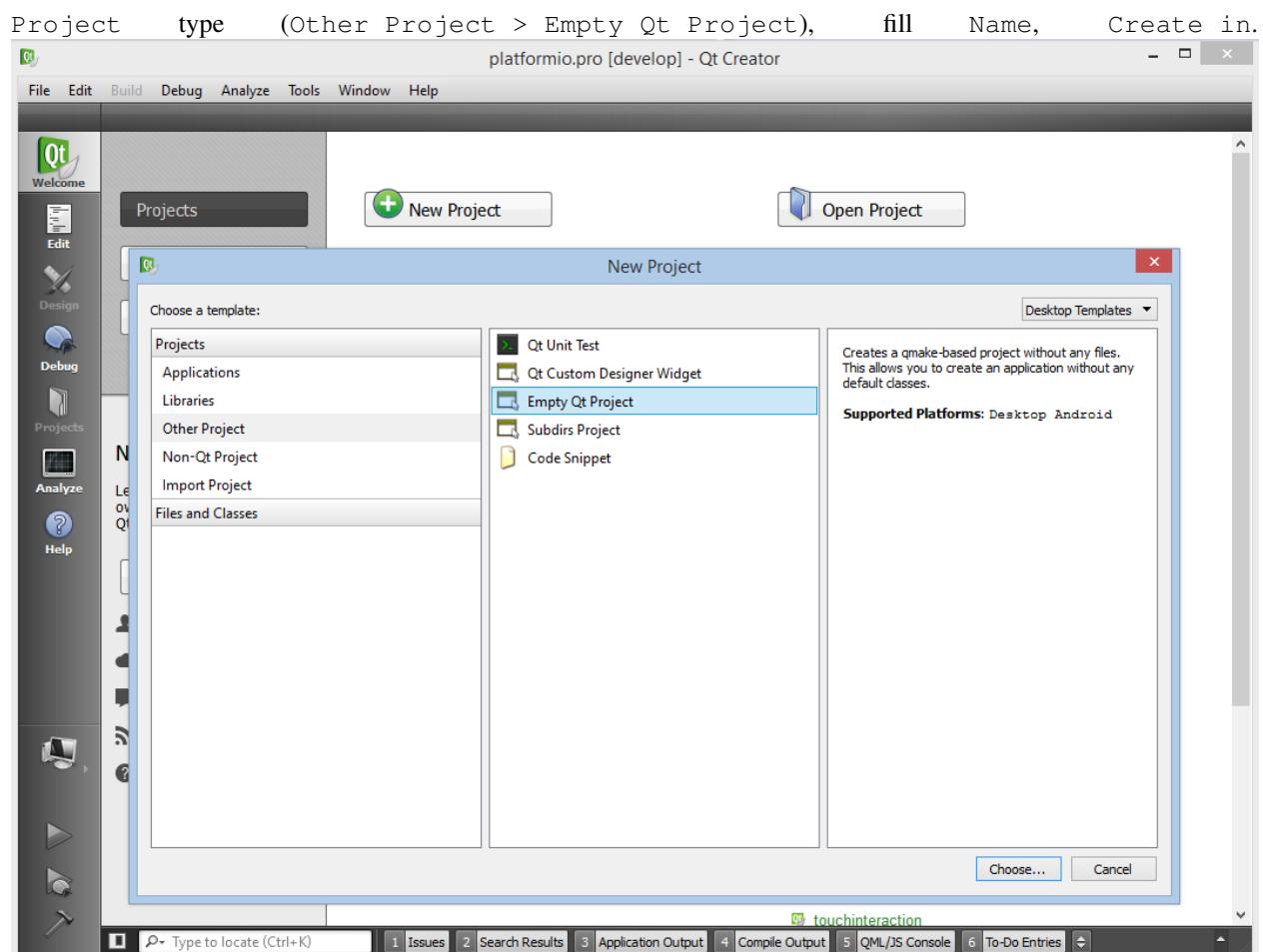


Warning: The libraries which are added, installed or used in the project after generating process wont be reflected in IDE. To fix it you need to reinitialize project using [platformio init](#) (repeat it).

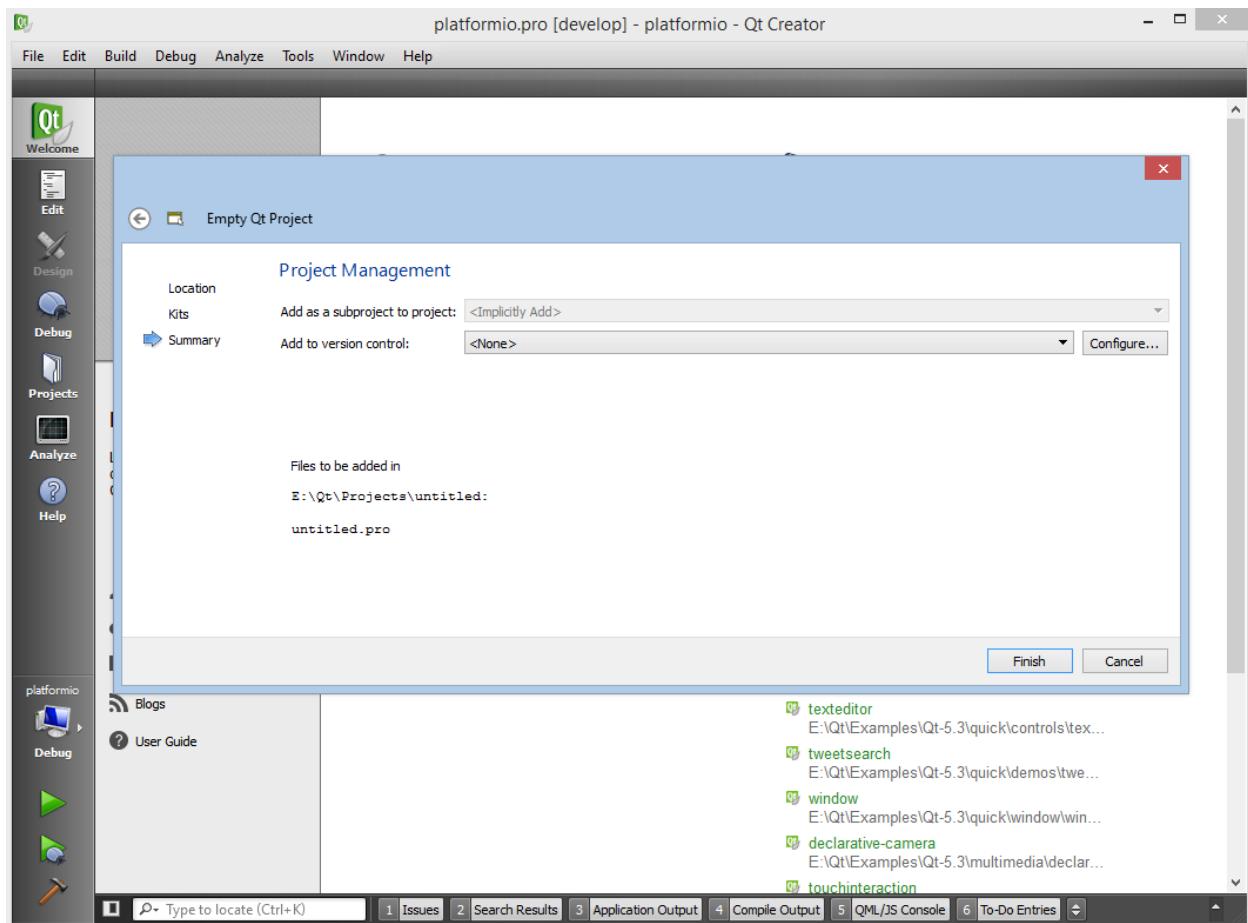
Manual Integration

Setup New Project

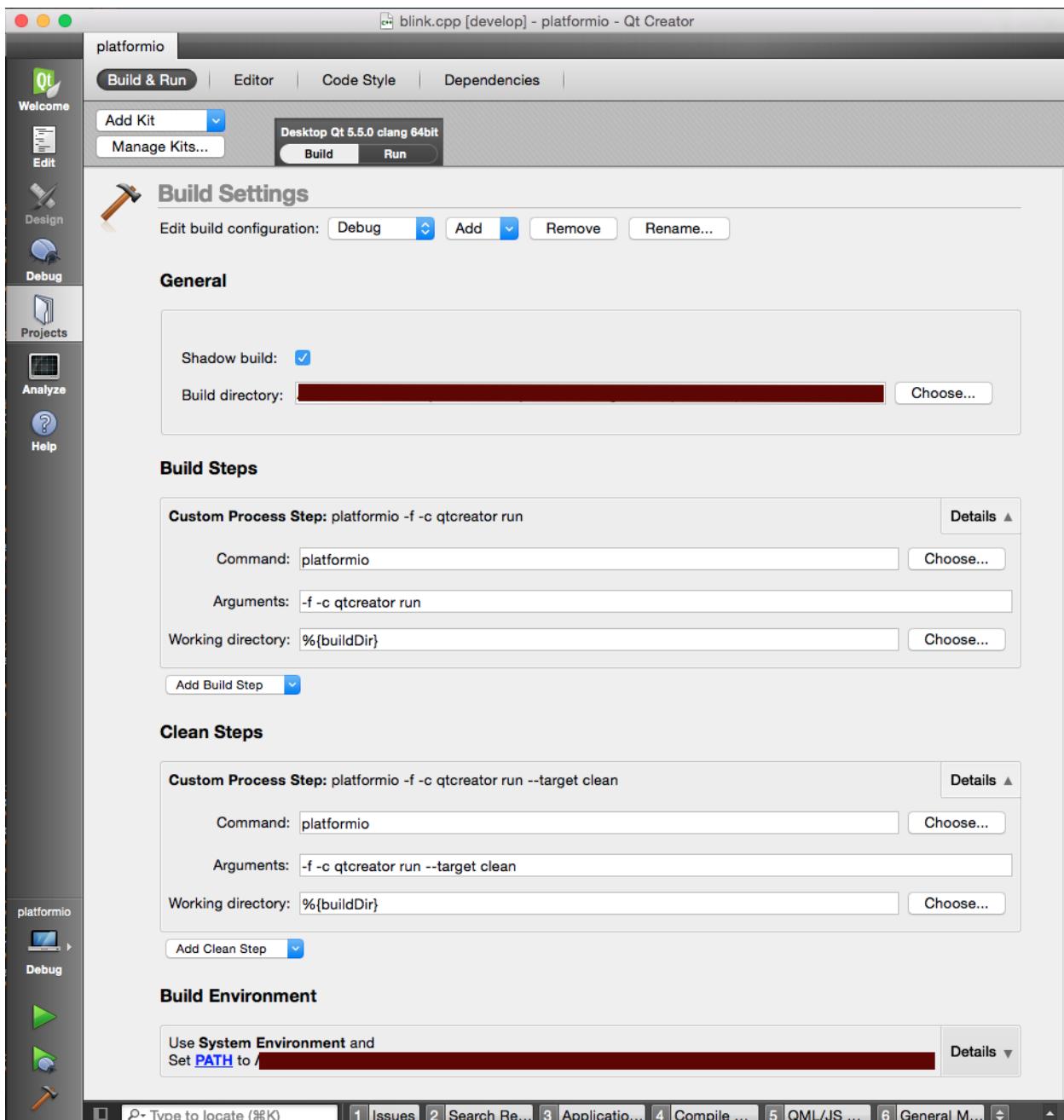
First of all, let's create new project from Qt Creator Start Page: New Project or using Menu: File > New File or Project, then select project with Empty Qt



On the next steps select any available kit and click Finish button.



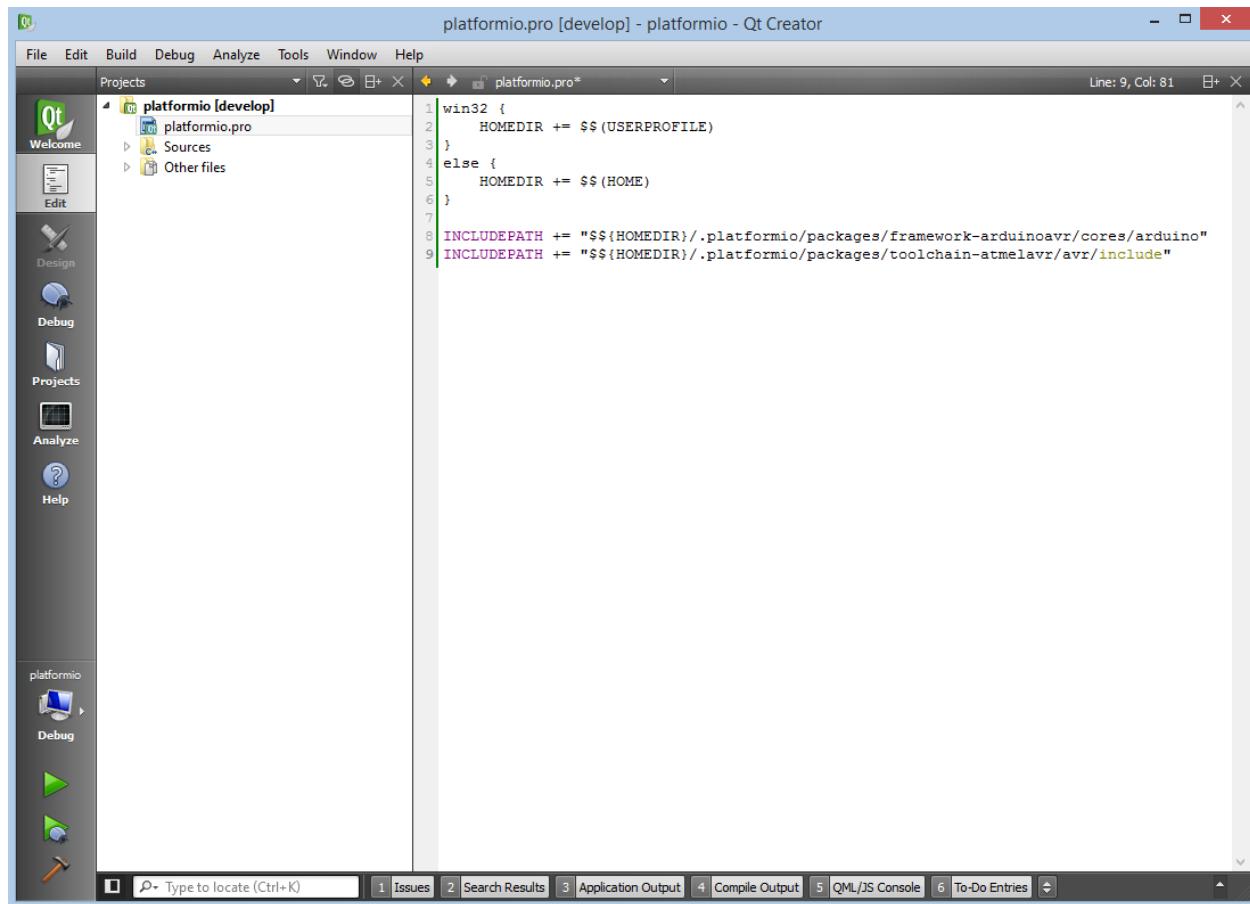
Secondly, we need to delete default build and clean steps and configure project with PlatformIO Build System (click on Projects label on left menu or **Ctrl+5** shortcut):



Thirdly, change project file by adding path to directories with header files. Please edit project file to match the following contents:

```
win32 {
    HOMEDIR += $$ (USERPROFILE)
}
else {
    HOMEDIR += $$ (HOME)
}

INCLUDEPATH += "$$ {HOMEDIR}/.platformio/packages/framework-arduinoavr/cores/arduino"
INCLUDEPATH += "$$ {HOMEDIR}/.platformio/packages/toolchain-atmelavr/avr/include"
```

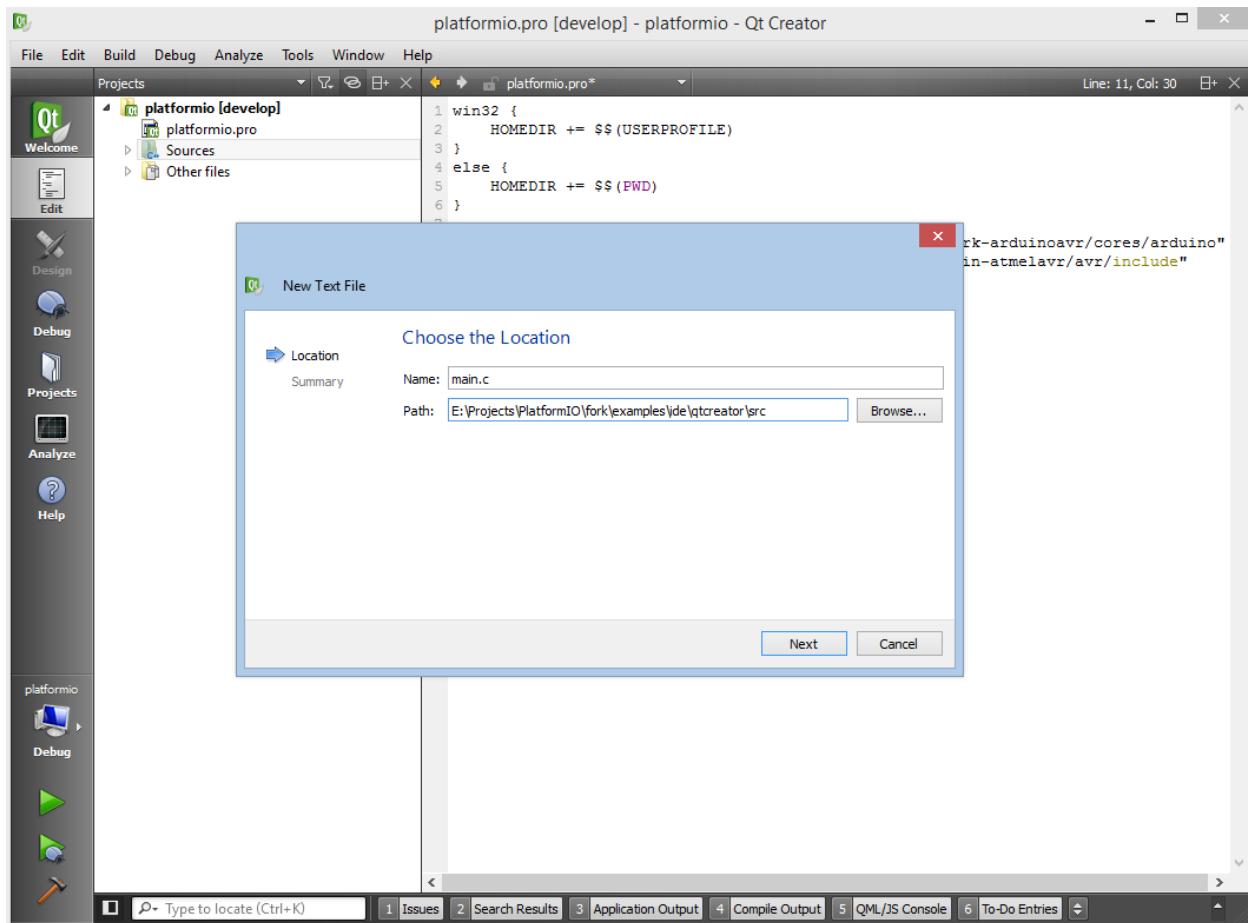


First program in Qt Creator

Simple “Blink” project will consist from two files:

1. In the console, navigate to the root of your project folder and initialize platformio project with `platformio init`
2. The main “C” source file named `main.c` must be located in the `src` directory.

Let’s create new text file named `main.c` using Menu: New File or Project > General > Text File:



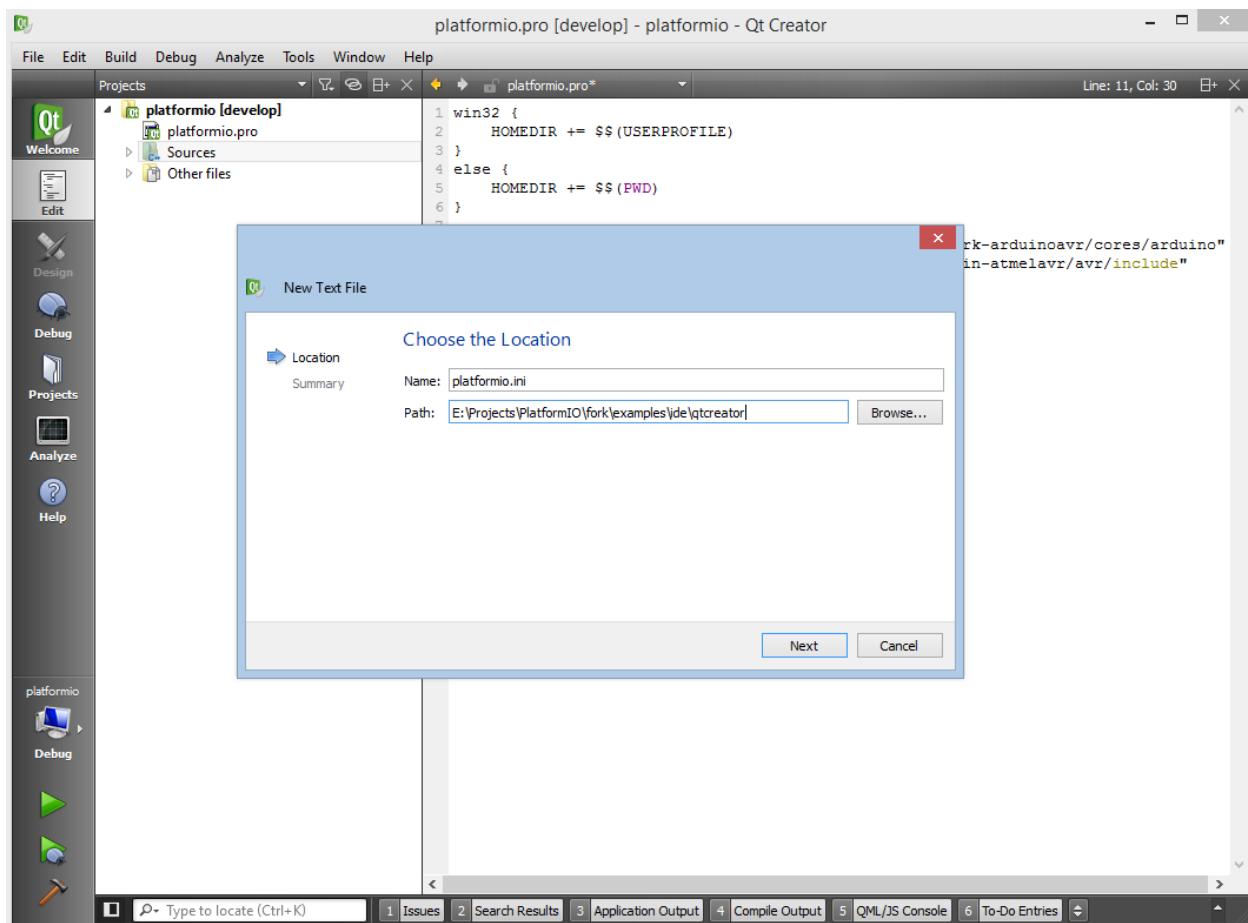
Copy the source code which is described below to file `main.c`.

```
#include "Arduino.h"
#define WLED    13 // Most Arduino boards already have an LED attached to pin 13 on
//the board itself

void setup()
{
  pinMode(WLED, OUTPUT); // set pin as output
}

void loop()
{
  digitalWrite(WLED, HIGH); // set the LED on
  delay(1000); // wait for a second
  digitalWrite(WLED, LOW); // set the LED off
  delay(1000); // wait for a second
}
```

3. Locate the project configuration file named `platformio.ini` at the root of the project directory and open it.



Edit the content to match the code described below.

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

[env:arduino_uno]
platform = atmelavr
framework = arduino
board = uno
```

Conclusion

Taking everything into account, we can build project with shortcut **Ctrl+Shift+B** or using Menu: **Build > Build All**.

Sublime Text

The Sublime Text is a cross-platform text and source code editor, with a Python application programming interface (API). Sublime Text is proprietary software. Its functionality is extendable with plugins. Most of the extending packages have free-software licenses and are community-built and maintained. Sublime Text lacks graphical setting dialogues and is entirely configured by editing text files.

Refer to the Sublime Text Documentation page for more detailed information.

The screenshot shows the PlatformIO IDE interface with several windows open:

- Project Explorer (Left):** Shows the project structure with files like main.cpp, USBCore.cpp, wiring_digital.c, and various GDB-related files.
- Code Editor (Top Left):** Displays the wiring_digital.c source code, specifically the digitalWrite function implementation.
- GDB Disassembly (Top Right):** Shows the assembly code corresponding to the C code in the editor.
- GDB Callstack (Bottom Left):** Lists the call stack frames.
- GDB Breakpoints (Bottom Middle):** Lists the current breakpoints.
- GDB Threads (Bottom Right):** Lists the threads and their states.
- GDB Console (Bottom Far Right):** Provides a terminal for interacting with the debugger.

Contents

- *Deviot Plugin*
 - *Integration*
 - *Project Generator*
 - *Manual Integration*
 - * *Initial configuration*
 - *Command Hotkeys*
 - * *First program in Sublime Text*

- * Conclusion
- Debugging

Deviot Plugin

We are glad to inform you about an awesome Sublime Text plugin for IoT development named [Deviot](#). It is based on [PlatformIO Core](#) and will automatically install it for you. Please visit [official Deviot page](#) for the further installation steps and documentation.

Integration

Project Generator

Integration process consists of these steps:

1. Open system Terminal and install [PlatformIO Core](#)
2. Create new folder for your project and change directory (`cd`) to it
3. Generate a project using PIO Core Project Generator (`platformio init --ide`)
4. Import project in IDE.

Choose board ID using [platformio boards](#) or Embedded Boards Explorer command and generate project via `platformio init --ide` command:

```
platformio init --ide sublimetext --board <ID>

# For example, generate project for Arduino Uno
platformio init --ide sublimetext --board uno
```

Then:

1. Import project via Menu: Project > Open Project... and select `platformio.sublime-project` from the folder where is located [Project Configuration File `platformio.ini`](#)
2. Select PlatformIO as build system: Menu: Tools > Build System > PlatformIO
3. Open source file from `src` directory (`*.c`, `*.cpp`, `*.ino`, etc.)
4. Build project: Menu: Tools > Build.

Also, you can access to all pre-configured targets via Menu: Tools > Builds With... (ST3)

- PlatformIO - Build - Build project without auto-uploading
- PlatformIO - Clean - Clean compiled objects.
- PlatformIO - Test - [PIO Unit Testing](#)
- PlatformIO - Upload - Build and upload (if no errors)
- PlatformIO - Upload using Programmer see [Upload using Programmer](#)
- PlatformIO - Upload SPIFFS image see [Uploading files to file system SPIFFS](#)
- PlatformIO - Update platforms and libraries - Update installed platforms and libraries via `platformio update`.

Manual Integration

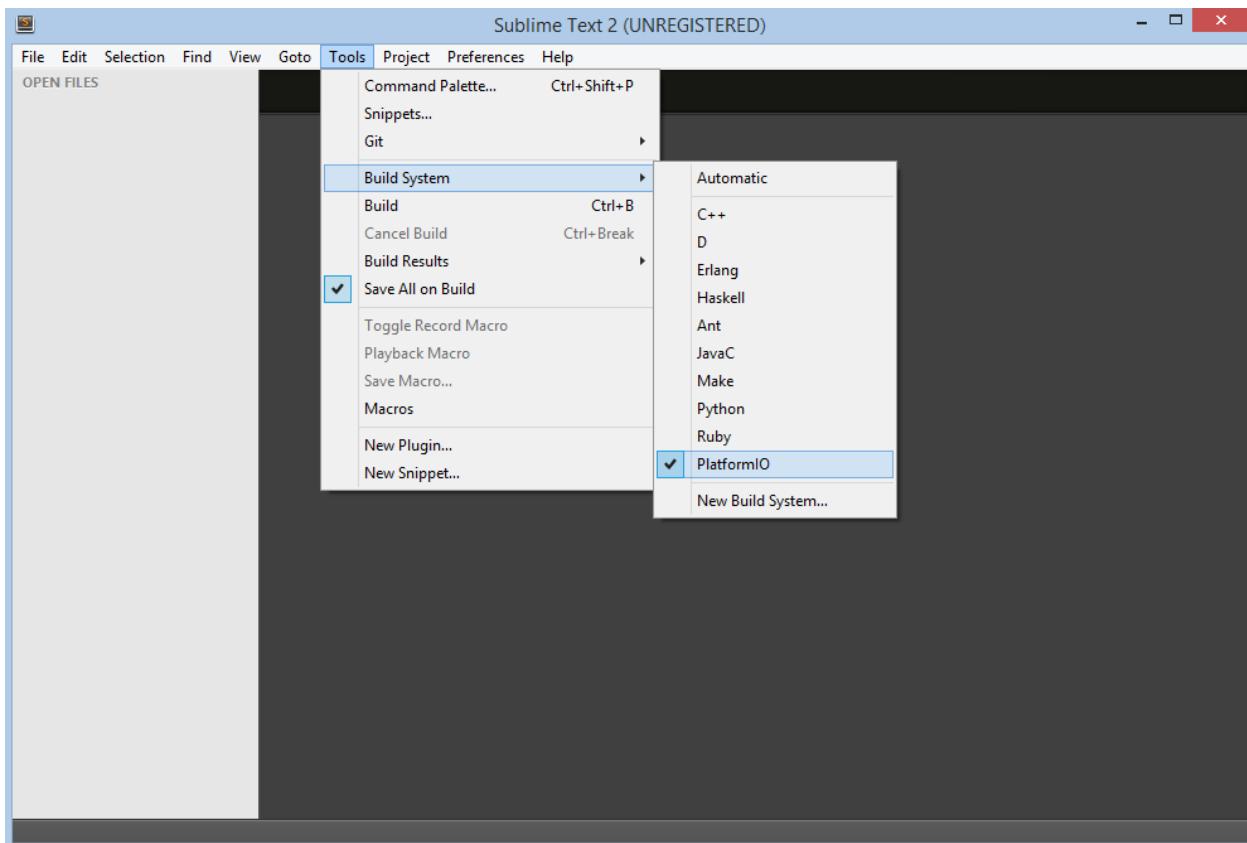
Note: Please verify that folder where is located platformio program is added to [PATH \(wiki\)](#) environment variable.

Initial configuration

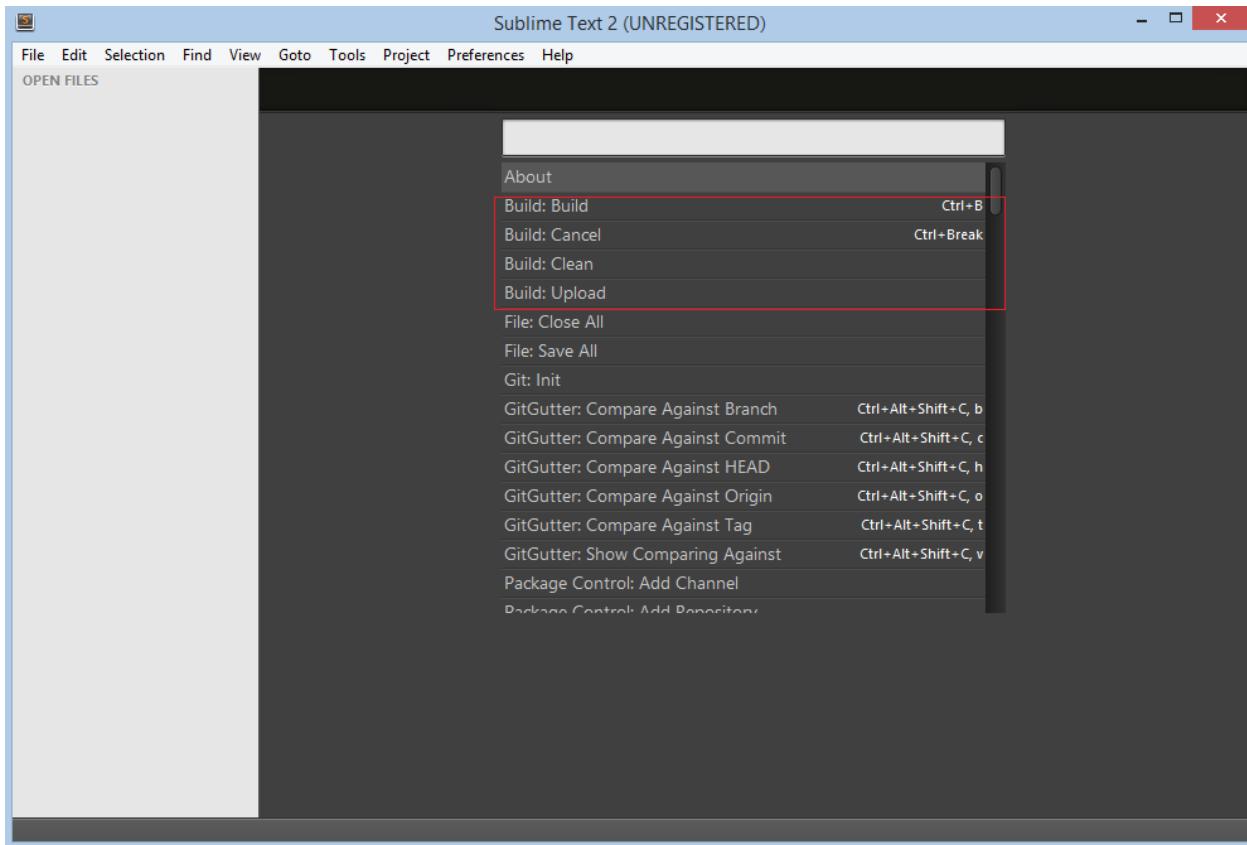
First of all, we need to create “New Build System” with name “PlatformIO” from Menu: Tools > Build System > New Build System and fill it like described below:

```
{
    "cmd": ["platformio", "-f", "-c", "sublimetext", "run"],
    "working_dir": "${project_path:${folder}}",
    "variants":
    [
        {
            "name": "Clean",
            "cmd": ["platformio", "-f", "-c", "sublimetext", "run", "--target", ↵"clean"]
        },
        {
            "name": "Upload",
            "cmd": ["platformio", "-f", "-c", "sublimetext", "run", "--target", ↵"upload"]
        }
    ]
}
```

Secondly, we need to select “PlatformIO” Build System from a list:

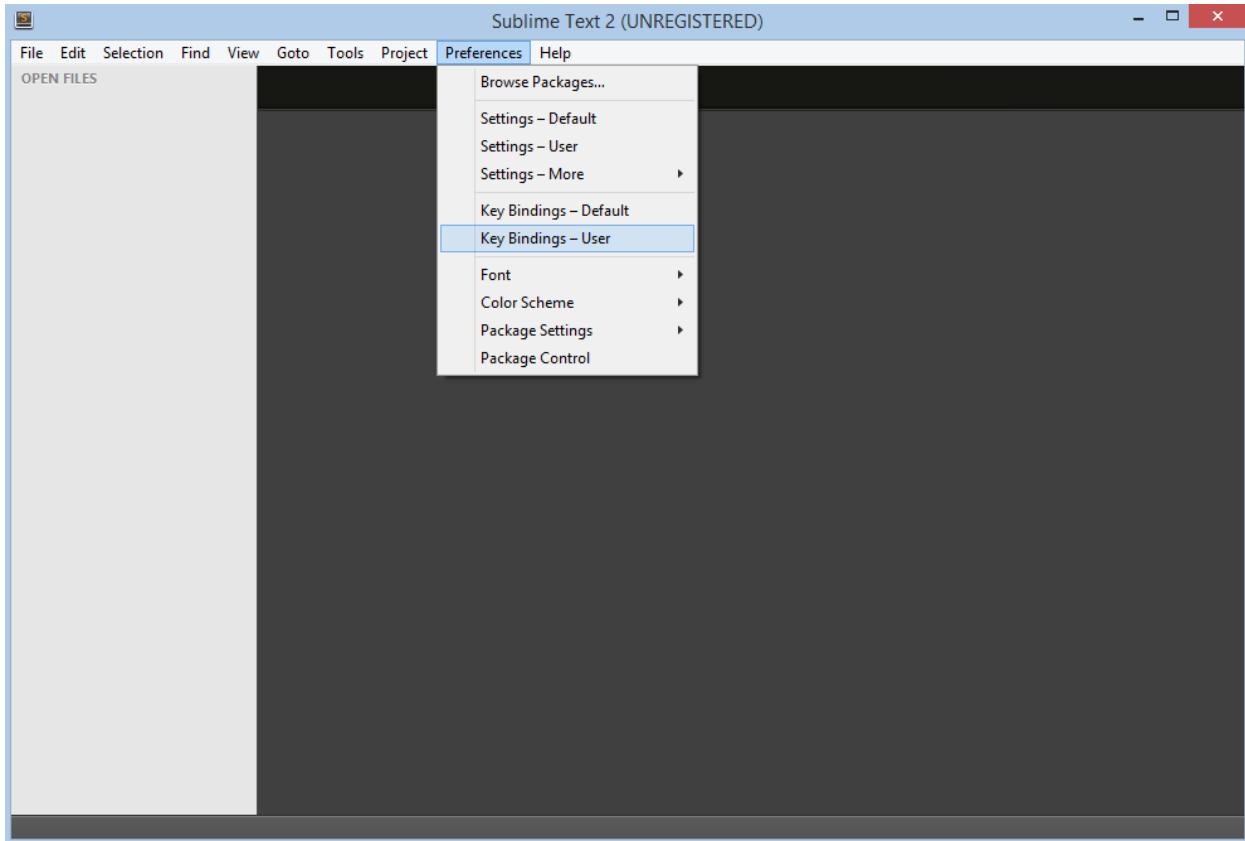


After that, we can use the necessary commands from Menu: Tools > Command Palette or with Ctrl+Shift+P (Windows/Linux) Cmd+Shift+P (Mac) shortcut.



Command Hotkeys

Sublime Text allows to bind own hotkey per command. Let's setup them for PlatformIO commands using shortcut Menu: Preferences > Key-Bindings – User:



We are going to use these shortcuts:

- F11 for clean project
- F12 for upload firmware to target device

In this case, the final code will look like:

```
[  
    { "keys": ["f11"], "command": "build", "args": {"variant": "Clean"} },  
    { "keys": ["f12"], "command": "build", "args": {"variant": "Upload"} }  
]
```

First program in Sublime Text

Simple “Blink” project will consist from two files:

1. Main “C” source file named `main.c` must be located in the `src` directory. Let’s create new file named `main.c` using Menu: File > New File or shortcut Ctrl+N (Windows/Linux) Cmd+N (Mac) with the next contents:

```
#include "Arduino.h"  
#define WLED 13 // Most Arduino boards already have an LED attached to pin 13 on  
// the board itself  
  
void setup()  
{  
    pinMode(WLED, OUTPUT); // set pin as output  
}
```

(continues on next page)

(continued from previous page)

```
void loop()
{
    digitalWrite(WLED, HIGH); // set the LED on
    delay(1000); // wait for a second
    digitalWrite(WLED, LOW); // set the LED off
    delay(1000); // wait for a second
}
```

2. Project Configuration File named `platformio.ini` must be located in the project root directory. Copy the source code which is described below to it.

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

[env:arduino_uno]
platform = atmelavr
framework = arduino
board = uno
```

Conclusion

Taking everything into account, we can open project directory in Sublime Text using Menu: File > Open Folder and build it with shortcut Ctrl+B (Windows/Linux) or Cmd+B (Mac), clean project with shortcut F11 and upload firmware to target with shortcut F12.

Debugging

A debugging feature is provided by *PIO Unified Debugger* and new debug configuration named “PlatformIO Debugger” is created. No need to do extra configuration steps!

1. Install `SublimeGDB` package
2. Launch debugger with F5
3. Wait for a while, PlatformIO will prepare project for debugging and session will be started soon.

VIM

`VIM` is an open-source, powerful and configurable text editor. Vim is designed for use both from a command-line interface and as a standalone application in a graphical user interface.

The screenshot shows a VIM editor window with three tabs:

- Makefile (~/Downloads/PlatformIO) - VIM**: Contains C code for an Arduino sketch. It includes a `setup()` function that sets pin 13 as an output and a `loop()` function that alternates the LED between HIGH and LOW states every second.
- ~/Downloads/PlatformIO/main.c**: Contains a PlatformIO environment configuration. It defines the platform as `atmelavr`, framework as `arduino`, and board as `pro16MHzatmega168`. It also specifies the device port as `/dev/ttyUSB0`.
- ~/Downloads/PlatformIO/platformio.ini**: Contains a PlatformIO project configuration. It sets the shell to `/bin/bash` and the PATH to `/usr/local/bin:$PATH`. It defines two targets: `all` (which runs `platformio run -t upload`) and `clean` (which runs `platformio run -t clean`).

The status bar at the bottom indicates the file type as `c`, encoding as `utf-8[unix]`, and current line as `11: 2`. The bottom right corner shows the file size as `73%` and the total number of lines as `2`.

Contents

- *VIM*
 - *Integration*
 - * “*Neomake-PlatformIO*” *Plugin*
 - * *Project Generator*
 - *Articles / Manuals*

Integration

Integration process consists of these steps:

1. Open system Terminal and install *PlatformIO Core*
2. Create new folder for your project and change directory (`cd`) to it
3. Generate a project using PIO Core Project Generator (`platformio init --ide`)
4. Import project in IDE.

“Neomake-PlatformIO” Plugin

Please visit [neomake-platformio](#) for the further installation steps and documentation.

Project Generator

Choose board ID using *platformio boards* or Embedded Boards Explorer command and generate project via *platformio init --ide* command:

```
platformio init --ide vim --board <ID>
```

Recommended bundles:

- Syntax highlight - Arduino-syntax-file
- Code Completion - YouCompleteMe (see configuration example by [Anthony Ford PlatformIO/YouCompleteMe Integration](#))
- Syntax checking - Syntastic

Put to the project directory `Makefile wrapper` with contents:

```
# Uncomment lines below if you have problems with $PATH
#SHELL := /bin/bash
#PATH := /usr/local/bin:$PATH

all:
    platformio -f -c vim run

upload:
    platformio -f -c vim run --target upload

clean:
    platformio -f -c vim run --target clean

program:
    platformio -f -c vim run --target program

uploadfs:
    platformio -f -c vim run --target uploadfs

update:
    platformio -f -c vim update
```

Pre-defined targets:

- Build - Build project without auto-uploading
- Clean - Clean compiled objects.
- Upload - Build and upload (if no errors)
- Upload using Programmer see [Upload using Programmer](#)
- Upload SPIFFS image see [Uploading files to file system SPIFFS](#)
- Update platforms and libraries - Update installed platforms and libraries via [platformio update](#).

Now, in VIM `cd /path/to/this/project` and press `Ctrl+B` or `Cmd+B` (Mac). *PlatformIO* should compile your source code from the `src` directory, make firmware and upload it.

Note: If hotkey doesn't work for you, try to add this line `nnoremap <C-b> :make<CR>` to `~/.vimrc`

Articles / Manuals

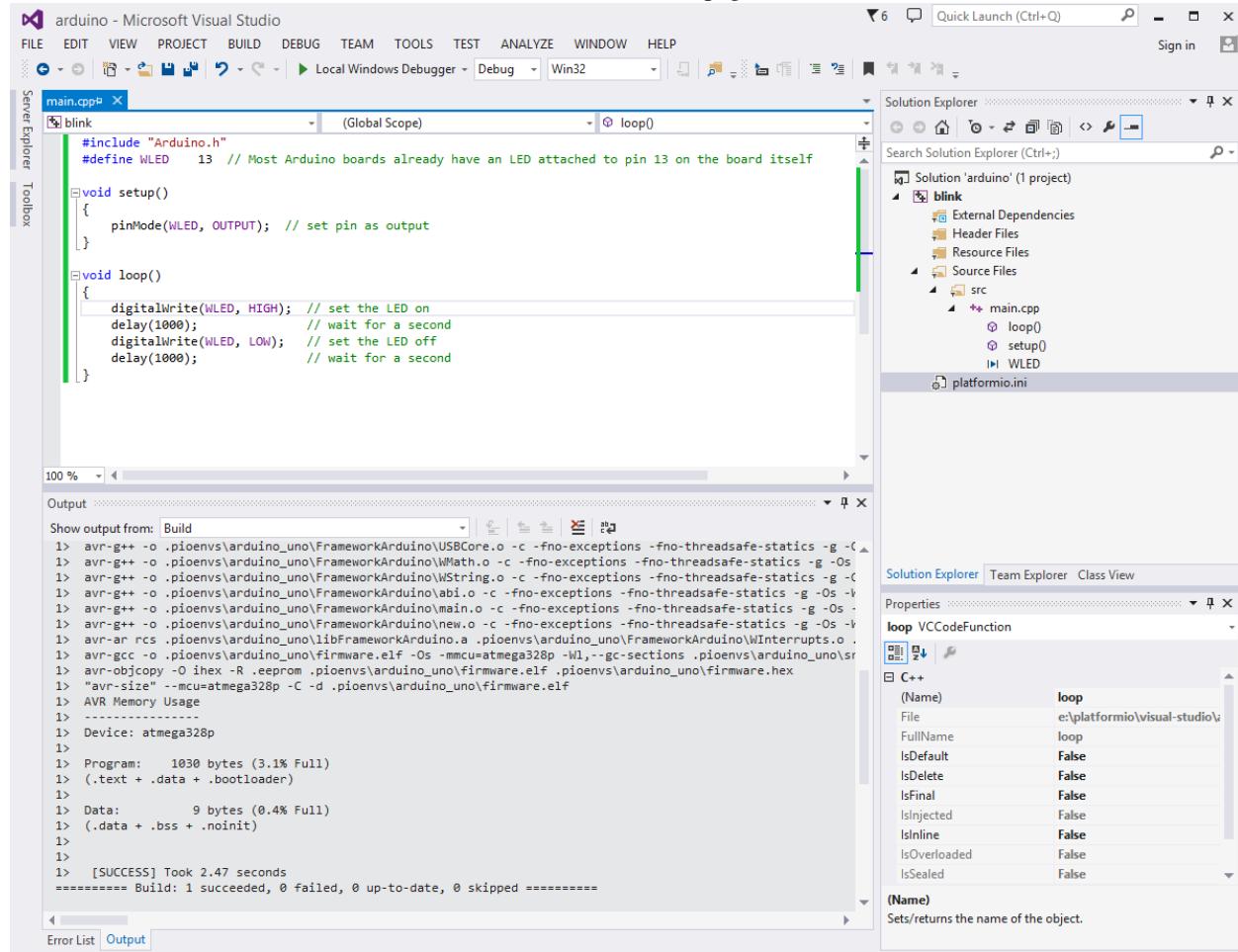
- Arduino : vim + platformio (Arduino development at the command line: VIM + PlatformIO, Japanese)

See a full list with [Articles about us](#).

Visual Studio

The Microsoft Visual Studio (Free) is an integrated development environment (IDE) from Microsoft. Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring.

Refer to the Visual Studio Documentation page for more detailed information.



Contents

- [Visual Studio](#)
 - [Integration](#)

- * *Project Generator*
- * *Manual Integration*
 - *Setup New Project*
 - *First program in Visual Studio*
 - *Conclusion*
- *Known issues*
- * *IntelliSense Errors*

Integration

Integration process consists of these steps:

1. Open system Terminal and install *PlatformIO Core*
 2. Create new folder for your project and change directory (`cd`) to it
 3. Generate a project using PIO Core Project Generator (`platformio init --ide`)
 4. Import project in IDE.
-

Project Generator

Choose board ID using *platformio boards* or Embedded Boards Explorer command and generate project via `platformio init --ide` command:

```
platformio init --ide sublimetext --board <ID>
# For example, generate project for Arduino Uno
platformio init --ide visualstudio --board uno
```

Then:

1. Import this project via Menu: File > Open > Project/Solution and specify root directory where is located *Project Configuration File platformio.ini*
2. Open source file from `src` directory (`*.c`, `*.cpp`, `*.ino`, etc.)
3. Build project: Menu: Build > Build Solution.

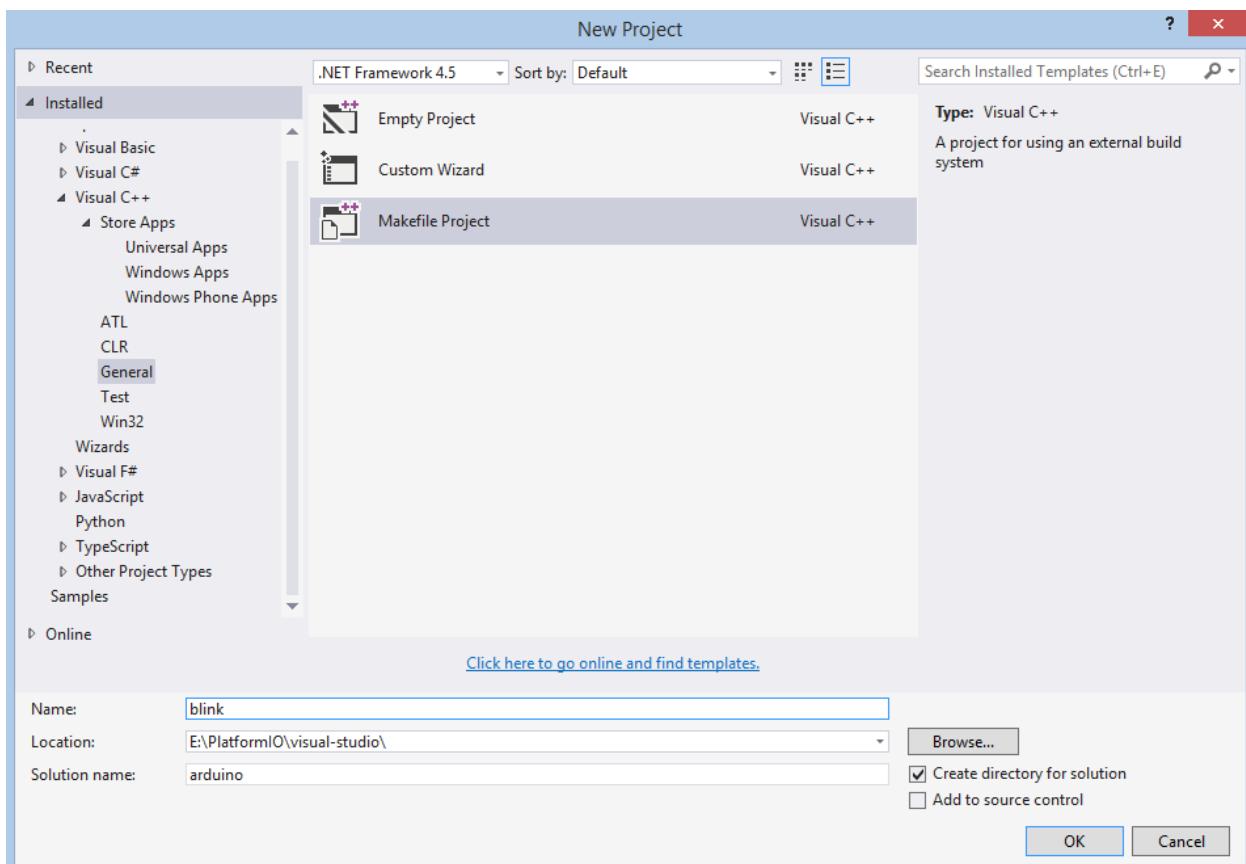
Warning: The libraries which are added, installed or used in the project after generating process wont be reflected in IDE. To fix it you need to reinitialize project using `platformio init` (repeat it).

Manual Integration

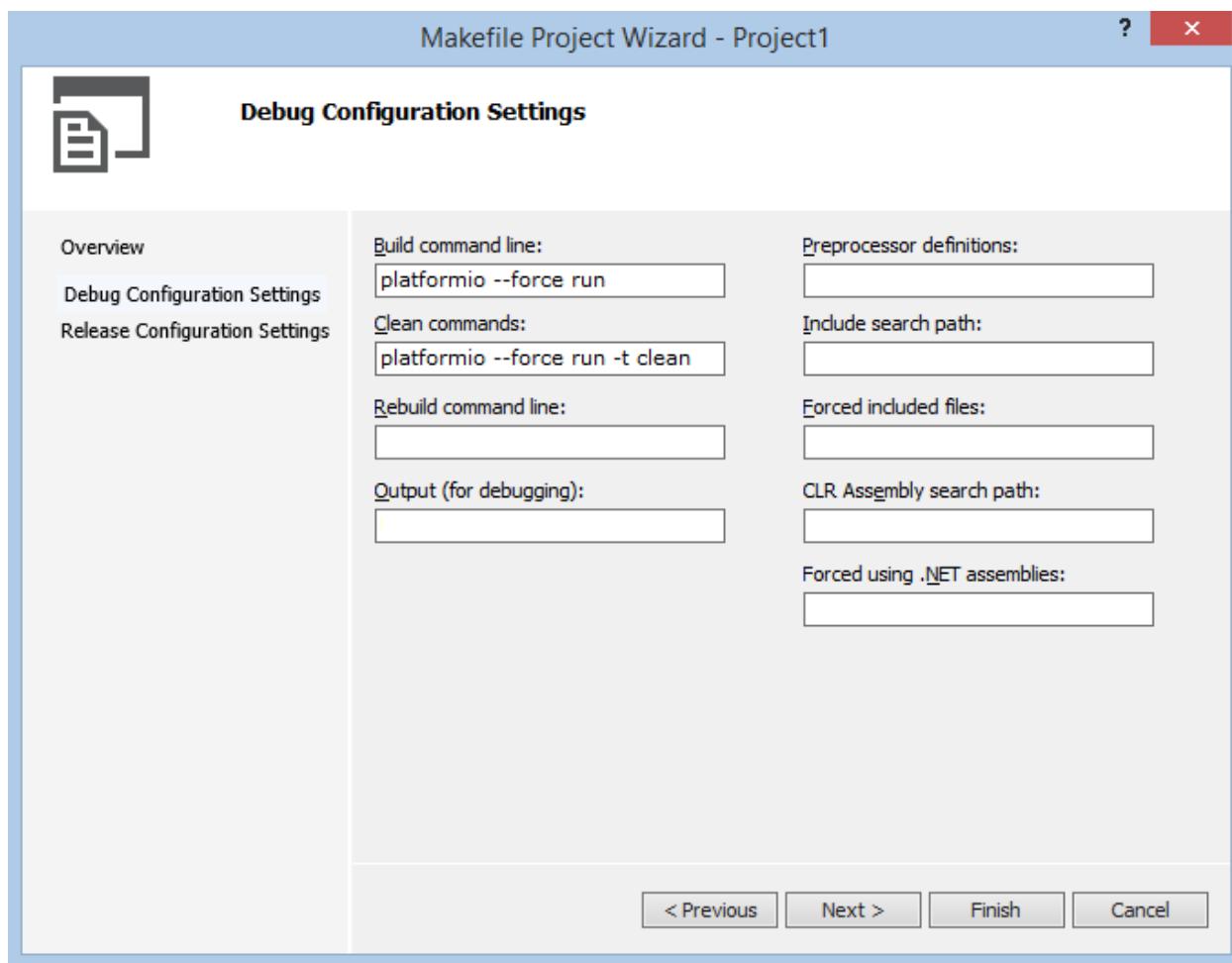
Setup New Project

First of all, let's create new project from Visual Studio Start Page: Start > New Project or using Menu: File > New > Project, then select project with Makefile type (Visual C++ > General >

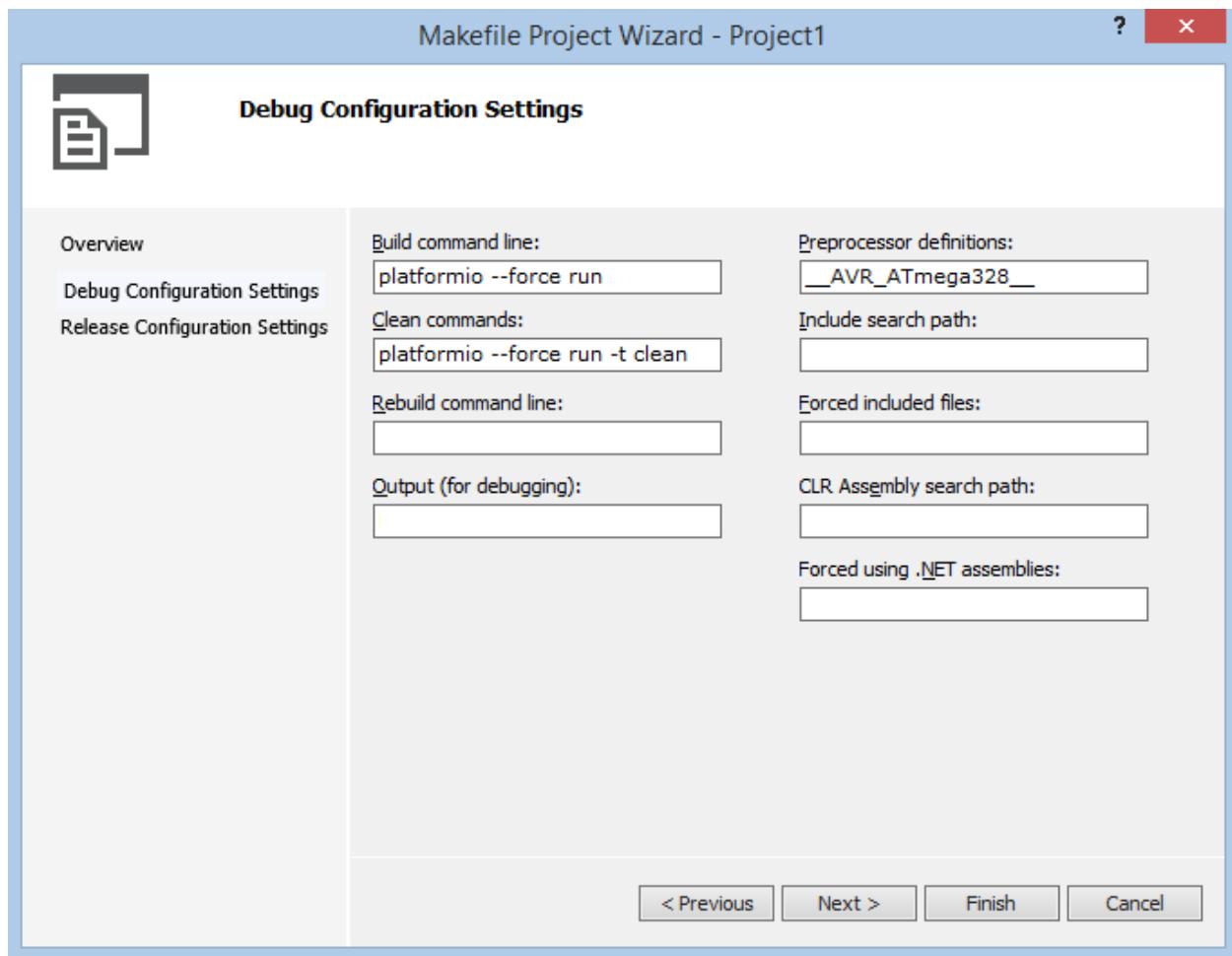
Makefile Project), fill Project name, Solution name, Location fields and press OK button.



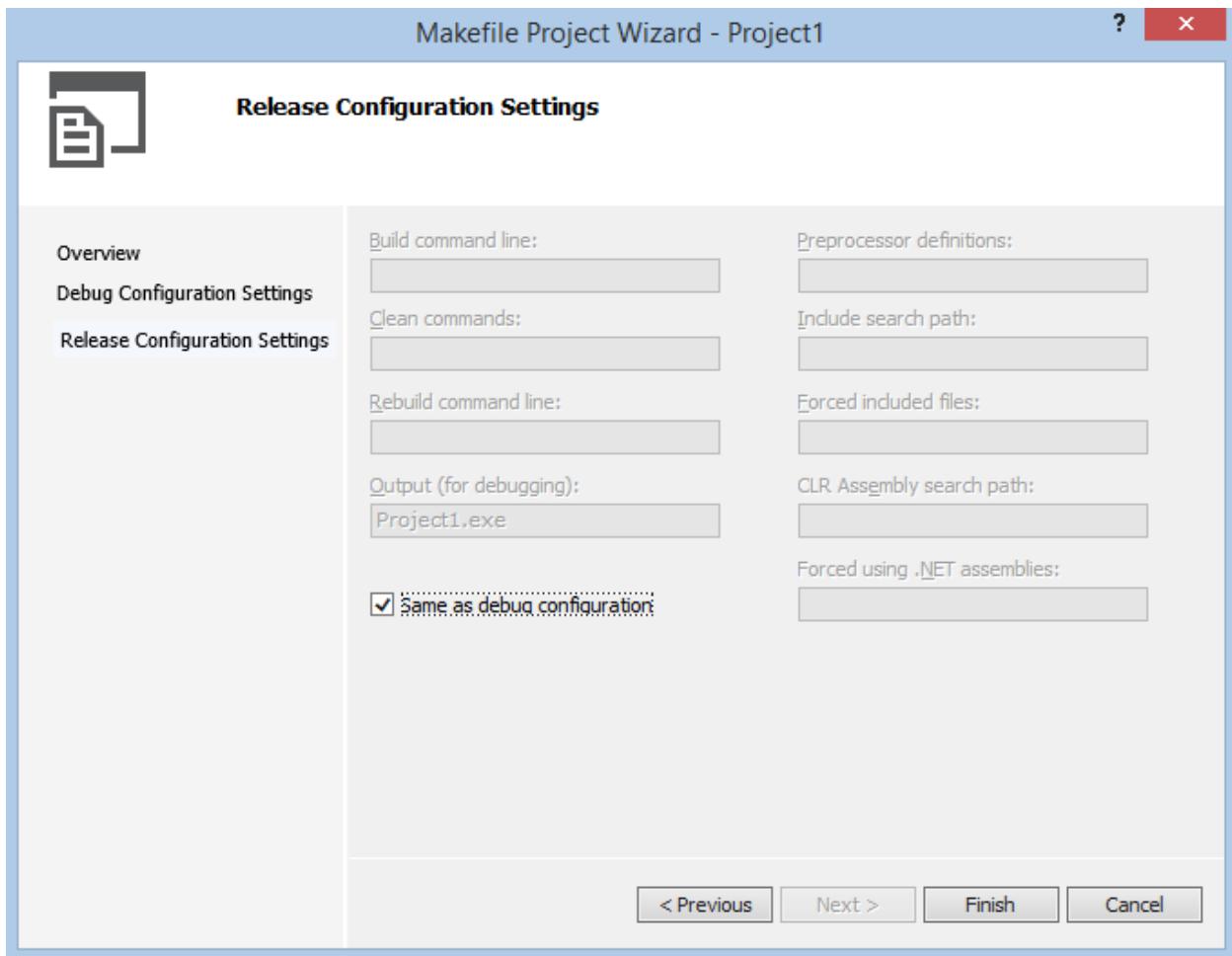
Secondly, we need to configure project with PlatformIO Build System:



If we want to use native AVR programming, we have to specify additional preprocessor symbol (“Preprocessor definitions” field) about your MCU. For example, an Arduino Uno is based on the ATmega328 MCU. In this case We will add new definition `__AVR_ATmega328__`.

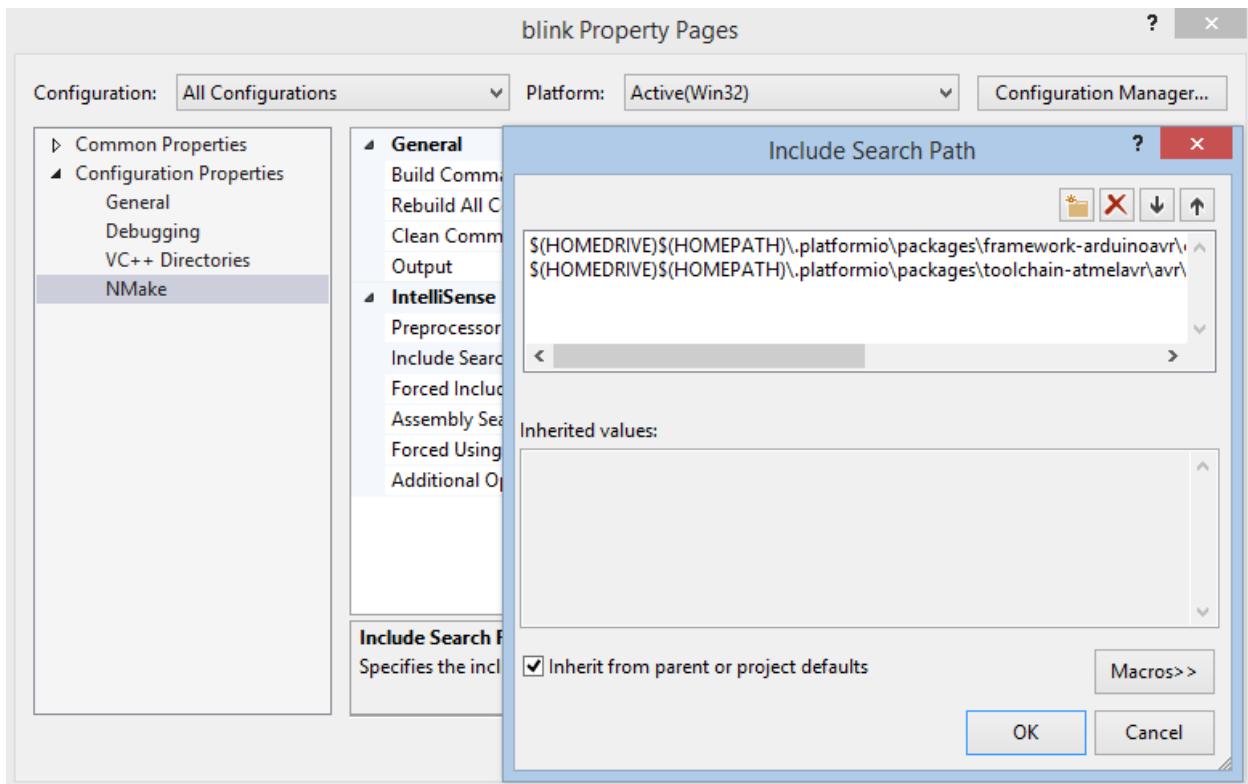


Release Configuration is the same as Debug, so on the next step we check “Same as Debug Configuration” and click “Finish” button.



Thirdly, we need to add directories with header files using project properties (right click on the project name or Alt+Enter shortcut) and add two directories to Configuration Properties > NMake > Include Search Path:

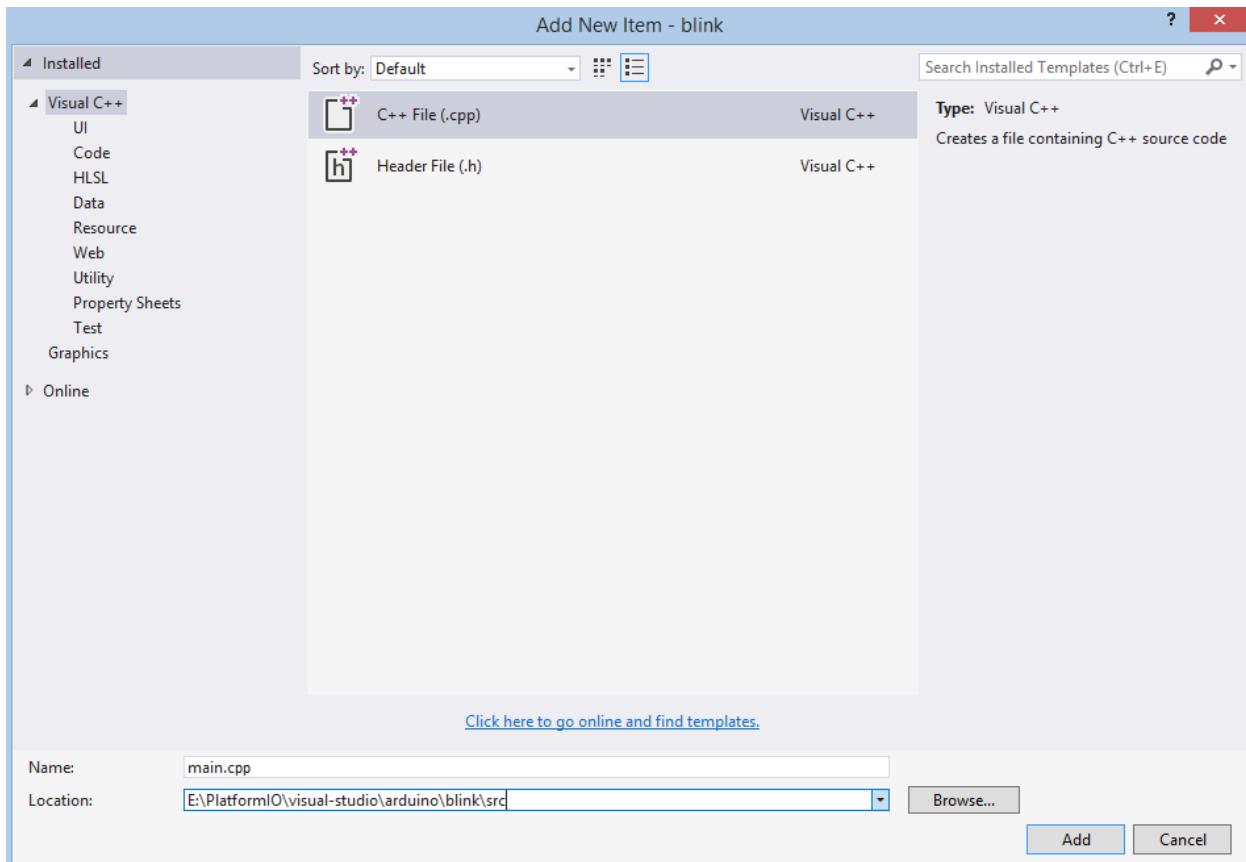
```
$ (HOMEDRIVE) $ (HOMEPATH) \.platformio\packages\toolchain-atmelavr\avr\include  
$ (HOMEDRIVE) $ (HOMEPATH) \.platformio\packages\framework-arduinoavr\cores\arduino
```



First program in Visual Studio

Simple “Blink” project will consist from two files:

1. Main “C++” source file named `main.cpp` must be located in the `src` directory. Let’s create new file named `main.cpp` using Menu: File > New File or shortcut `Ctrl+N`:



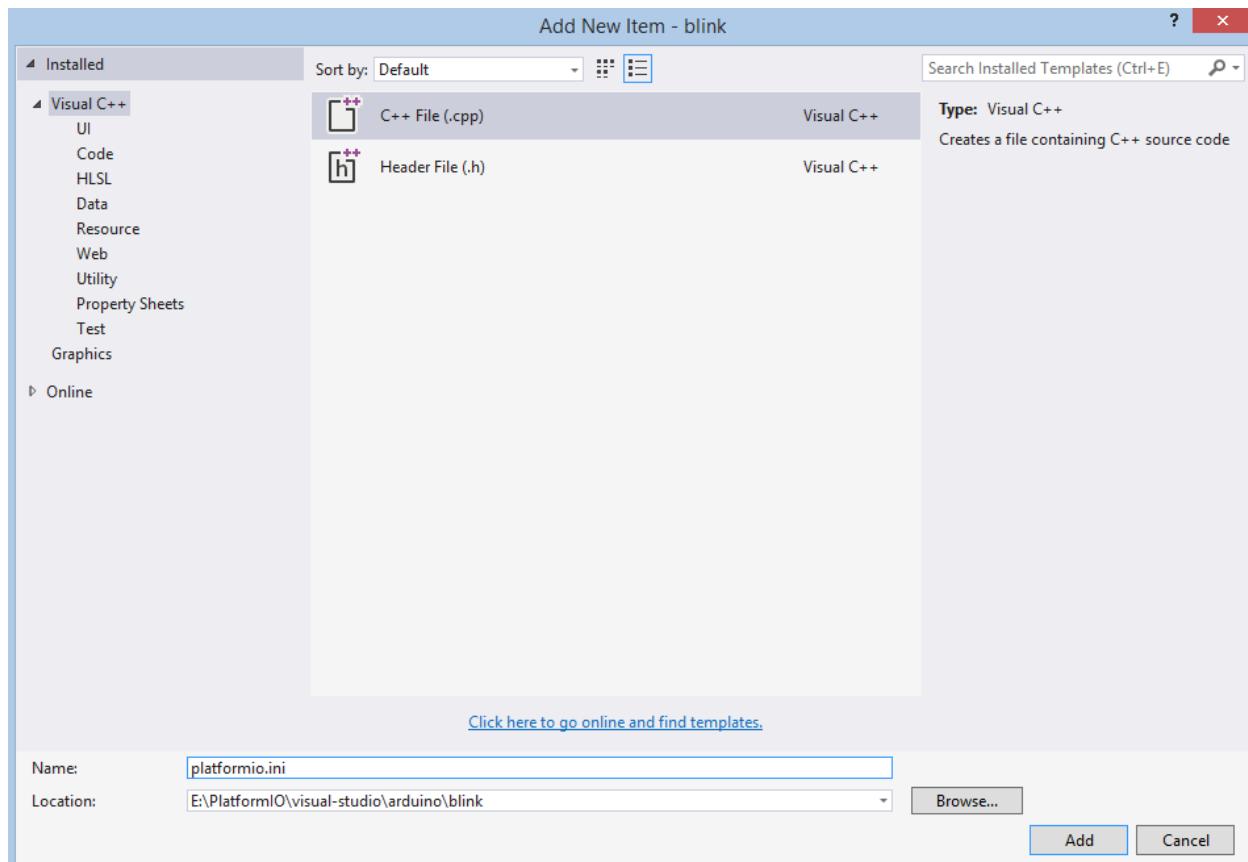
Copy the source code which is described below to file `main.cpp`.

```
#include "Arduino.h"
#define WLED 13 // Most Arduino boards already have an LED attached to pin 13 on_
             // the board itself

void setup()
{
    pinMode(WLED, OUTPUT); // set pin as output
}

void loop()
{
    digitalWrite(WLED, HIGH); // set the LED on
    delay(1000); // wait for a second
    digitalWrite(WLED, LOW); // set the LED off
    delay(1000); // wait for a second
}
```

2. Project Configuration File named `platformio.ini` must be located in the project root directory.



Copy the source code which is described below to it.

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

[env:arduino_uno]
platform = atmelavr
framework = arduino
board = uno
```

Conclusion

Taking everything into account, we can build project with shortcut **Ctrl+Shift+B** or using Menu: **Build > Build Solution**.

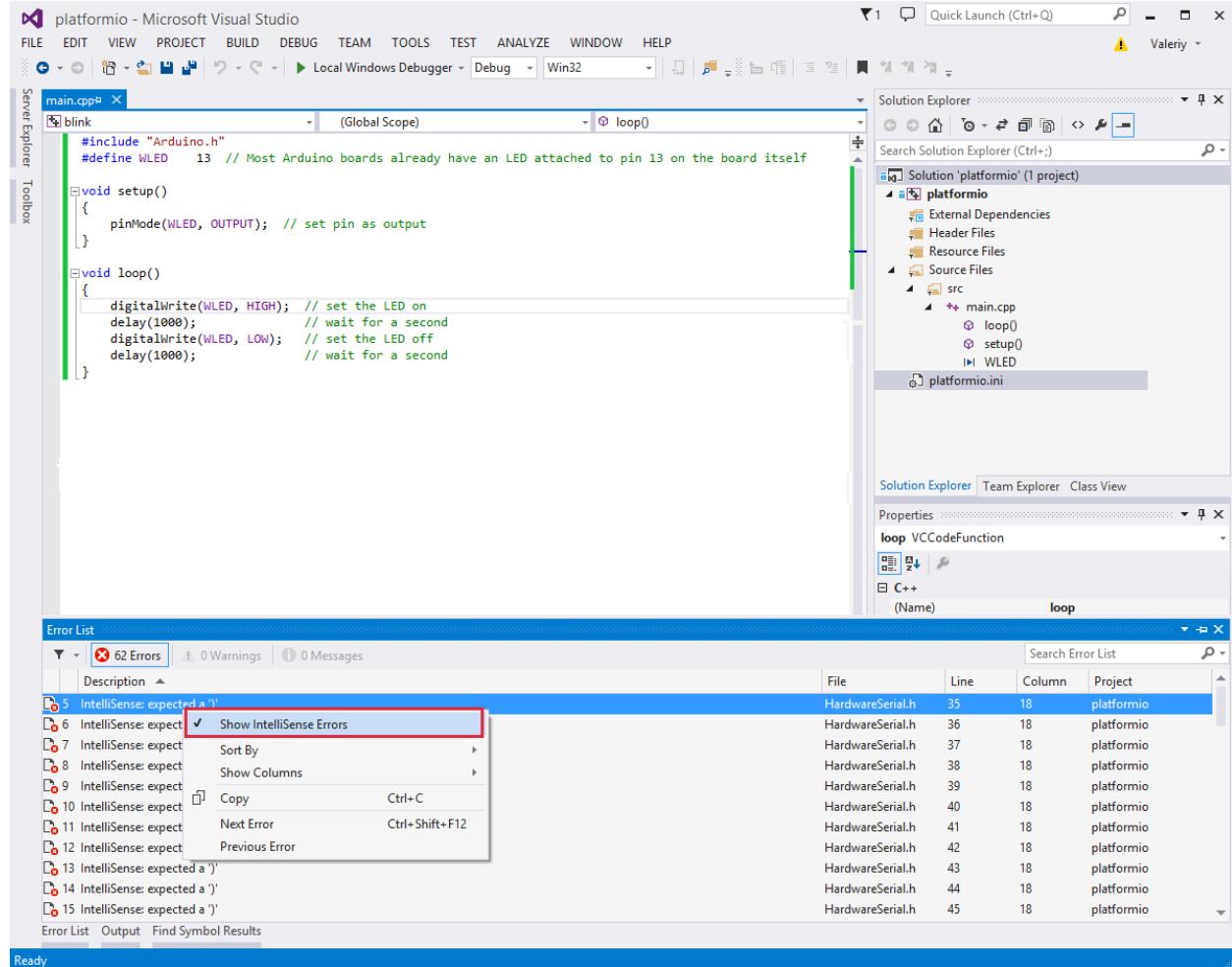
Known issues

IntelliSense Errors

VS Studio does not allow to specify for project other toolchain which will be used by IntelliSense. In this case, IntelliSense does not understand GCC-specific definitions.

However, these errors does not have any influence on PlatformIO Build System. It means that you can ignore them and rely on PlatformIO Build System messages which will be shown in output console after build.

Nevertheless, you can provide an IntelliSense-friendly definition of problematic GCC constructs and make sure that the GCC will ignore such definitions or disable IntelliSense error reporting at all. See details in [issue #543](#)



1.19 Continuous Integration

Continuous Integration (CI, [wiki](#)) is the practice, in software engineering, of merging all developer working copies with a shared mainline several times a day.

`platformio ci` command is intended to be used in combination with the build servers and the popular [Continuous Integration Software](#).

By integrating regularly, you can detect errors quickly, and locate them more easily.

1.19.1 AppVeyor

AppVeyor is an open-source hosted, distributed continuous integration service used to build and test projects hosted at GitHub on Windows family systems.

AppVeyor is configured by adding a file named `appveyor.yml`, which is a [YAML](#) format text file, to the root directory of the GitHub repository.

AppVeyor automatically detects when a commit has been made and pushed to a repository that is using AppVeyor, and each time this happens, it will try to build the project using `platformio ci` command. This includes commits to all branches, not just to the master branch. AppVeyor will also build and run pull requests. When that process has completed, it will notify a developer in the way it has been configured to do so — for example, by sending an email containing the build results (showing success or failure), or by posting a message on an IRC channel. It can be configured to build project on a range of different [Development Platforms](#).

Contents

- [AppVeyor](#)
 - [Integration](#)
 - [Examples](#)

Integration

Put `appveyor.yml` to the root directory of the GitHub repository.

```
build: off
environment:

matrix:
  - PLATFORMIO_CI_SRC: "path\\to\\source\\file.c"
  - PLATFORMIO_CI_SRC: "path\\to\\source\\file.ino"
  - PLATFORMIO_CI_SRC: "path\\to\\source\\directory"

install:
  - cmd: git submodule update --init --recursive
  - cmd: SET PATH=%PATH%;C:\Python27\Scripts
  - cmd: pip install -U platformio

test_script:
  - cmd: platformio ci --board=<ID_1> --board=<ID_2> --board=<ID_N>
```

For more details as for PlatformIO build process please look into `platformio ci` command.

Examples

1. Integration for [USB_Host_Shield_2.0](#) project. The `appveyor.yml` configuration file:

```
build: off
environment:

matrix:
  - PLATFORMIO_CI_SRC: "examples\\Bluetooth\\PS3SPP\\PS3SPP.ino"
```

(continues on next page)

(continued from previous page)

```

- PLATFORMIO_CI_SRC: "examples\\p12303\\p12303_gps\\p12303_gps.ino"

install:
  - cmd: git submodule update --init --recursive
  - cmd: SET PATH=%PATH%;C:\Python27\Scripts
  - cmd: pip install -U platformio
  - cmd: git clone https://github.com/xxxajk/spi4teensy3.git C:\spi4teensy

test_script:
  - cmd: platformio ci --lib="." --lib="C:\\spi4teensy" --board=uno --
    ↪board=teensy31 --board=due

```

1.19.2 CircleCI

CircleCI is a hosted cloud platform that provides hosted continuous integration, deployment, and testing to GitHub repositories.

CircleCI is configured by adding a file named `circle.yml`, which is a [YAML](#) format text file, to the root directory of the GitHub repository.

CircleCI automatically detects when a commit has been made and pushed to a repository that is using CircleCI, and each time this happens, it will try to build the project using `platformio ci` command. This includes commits to all branches, not just to the master branch. CircleCI will also build and run pull requests. When that process has completed, it will notify a developer in the way it has been configured to do so — for example, by sending an email containing the build results (showing success or failure), or by posting a message on an IRC channel. It can be configured to build project on a range of different [Development Platforms](#).

Contents

- [CircleCI](#)
 - [Integration](#)
 - * [Project as a library](#)
 - * [Library dependencies](#)
 - [Install dependent library using Library Manager](#)
 - [Manually download dependent library and include in build process via --lib option](#)
 - * [Custom Build Flags](#)
 - * [Advanced configuration](#)
 - [Examples](#)

Integration

Please make sure to read CircleCI Getting Started guide first.

```

dependencies:
  pre:
    # Install the latest stable PlatformIO
    - sudo pip install -U platformio

```

(continues on next page)

(continued from previous page)

```
test:
  override:
    - platformio ci path/to/test/file.c --board=<ID_1> --board=<ID_2> --board=<ID_
  ↵N>
    - platformio ci examples/file.ino --board=<ID_1> --board=<ID_2> --board=<ID_N>
    - platformio ci path/to/test/directory --board=<ID_1> --board=<ID_2> --board=
  ↵<ID_N>
```

For more details as for PlatformIO build process please look into [platformio ci](#).

Project as a library

When project is written as a library (where own examples or testing code use it), please use `--lib=". "` option for [platformio ci](#) command

```
script:
  - platformio ci --lib=". " --board=<ID_1> --board=<ID_2> --board=<ID_N>
```

Library dependencies

There 2 options to test source code with dependent libraries:

Install dependent library using Library Manager

```
dependencies:
  pre:
    # Install the latest stable PlatformIO
    - sudo pip install -U platformio

    # OneWire Library with ID=1 https://platformio.org/lib/show/1/OneWire
    - platformio lib -g install 1

test:
  override:
    - platformio ci path/to/test/file.c --board=<ID_1> --board=<ID_2> --board=<ID_
  ↵N>
```

Manually download dependent library and include in build process via `--lib` option

```
dependencies:
  pre:
    # Install the latest stable PlatformIO
    - sudo pip install -U platformio

    # download library to the temporary directory
    - wget https://github.com/PaulStoffregen/OneWire/archive/master.zip -O /tmp/
  ↵onewire_source.zip
    - unzip /tmp/onewire_source.zip -d /tmp/
```

(continues on next page)

(continued from previous page)

```
test:
  override:
    - platformio ci path/to/test/file.c --lib="/tmp/OneWire-master" --board=<ID_1>
  ↵ --board=<ID_2> --board=<ID_N>
```

Custom Build Flags

PlatformIO allows to specify own build flags using `PLATFORMIO_BUILD_FLAGS` environment

```
machine:
  environment:
    PLATFORMIO_BUILD_FLAGS: -D SPECIFIC_MACROS -I/exra/inc
```

For the more details, please follow to [available build flags/options](#).

Advanced configuration

PlatformIO allows to configure multiple build environments for the single source code using [Project Configuration File `platformio.ini`](#).

Instead of `--board` option, please use `platformio ci --project-conf`

```
test:
  override:
    - platformio ci path/to/test/file.c --project-conf=/path/to/platformio.ini
```

Examples

1. Custom build flags

```
dependencies:
  cache_directories:
    - "~/.platformio"

pre:
  - sudo pip install -U platformio

  # pre-install PlatformIO development platforms, they will be cached
  - platformio platform install atmelavr atmelsam teensy

  #
  # Libraries from PlatformIO Library Registry:
  #
  # https://platformio.org/lib/show/416/TinyGPS
  # https://platformio.org/lib/show/417/SPI4Teensy3
  - platformio lib -g install 416 417

test:
  override:
    - platformio ci examples/acm/acm_terminal --board=uno --board=teensy31 --
  ↵board=due --lib=".">
```

(continues on next page)

(continued from previous page)

```

    - platformio ci examples/adk/adk_barcode --board=uno --board=teensy31 --
↳ board=due --lib="."
    - platformio ci examples/adk/ArduinoBlinkLED --board=uno --board=teensy31 --
↳ board=due --lib="."
    - platformio ci examples/adk/demokit_20 --board=uno --board=teensy31 --
↳ board=due --lib="."
    # ...
    - platformio ci examples/Xbox/XBOXUSB --board=uno --board=teensy31 --
↳ board=due --lib="."

```

2. Dependency on external libraries

```

dependencies:
  pre:
    # Install the latest stable PlatformIO
    - sudo pip install -U platformio

    # download dependent libraries
    - wget https://github.com/jcw/jeelib/archive/master.zip -O /tmp/jeelib.zip
    - unzip /tmp/jeelib.zip -d /tmp

    - wget https://github.com/Rodot/Gamebuino/archive/master.zip -O /tmp/
↳ gamebuino.zip
    - unzip /tmp/gamebuino.zip -d /tmp

test:
  override:
    - platformio ci examples/backSoon/backSoon.ino --lib=". --lib="/tmp/jeelib-
↳ master" --lib="/tmp/Gamebuino-master/libraries/tinyFAT" --board=uno --
↳ board=megaatmega2560
      - platformio ci examples/etherNode/etherNode.ino --lib=". --lib="/tmp/
↳ jeelib-master" --lib="/tmp/Gamebuino-master/libraries/tinyFAT" --board=uno --
↳ board=megaatmega2560
      - platformio ci examples/getDHCPandDNS/getDHCPandDNS.ino --lib=". --lib="/
↳ tmp/jeelib-master" --lib="/tmp/Gamebuino-master/libraries/tinyFAT" --board=uno --
↳ board=megaatmega2560
      - platformio ci examples/getStaticIP/getStaticIP.ino --lib=". --lib="/tmp/
↳ jeelib-master" --lib="/tmp/Gamebuino-master/libraries/tinyFAT" --board=uno --
↳ board=megaatmega2560
      # ...
      - platformio ci examples/twitter/twitter.ino --lib=". --lib="/tmp/jeelib-
↳ master" --lib="/tmp/Gamebuino-master/libraries/tinyFAT" --board=uno --
↳ board=megaatmega2560
      - platformio ci examples/udpClientSendOnly/udpClientSendOnly.ino --lib=". --
↳ lib="/tmp/jeelib-master" --lib="/tmp/Gamebuino-master/libraries/tinyFAT" --
↳ board=uno --board=megaatmega2560
      - platformio ci examples/udpListener/udpListener.ino --lib=". --lib="/tmp/
↳ jeelib-master" --lib="/tmp/Gamebuino-master/libraries/tinyFAT" --board=uno --
↳ board=megaatmega2560
      - platformio ci examples/webClient/webClient.ino --lib=". --lib="/tmp/
↳ jeelib-master" --lib="/tmp/Gamebuino-master/libraries/tinyFAT" --board=uno --
↳ board=megaatmega2560

```

- Configuration file: <https://github.com/ivankravets/ethercard/blob/master/circle.yaml>
- Build History: <https://circleci.com/gh/ivankravets/ethercard/tree/master>

1.19.3 Drone

Drone is a hosted continuous integration service. It enables you to conveniently set up projects to automatically build, test, and deploy as you make changes to your code to [GitHub](#) and [BitBucket](#) repositories.

Drone is configured by modifying settings in your project control panel.

Drone automatically detects when a commit has been made and pushed to a repository that is using Drone, and each time this happens, it will try to build the project using `platformio ci` command. This includes commits to all branches, not just to the master branch. Drone will also build and run pull requests. When that process has completed, it will notify a developer in the way it has been configured to do so — for example, by sending an email containing the build results (showing success or failure). It can be configured to build project on a range of different [Development Platforms](#).

Contents

- [*Drone*](#)
 - [*Integration*](#)
 - [*Examples*](#)

Integration

Please fill all fields for your project in the Drone control panel:

Environment Variables:

```
PLATFROMIO_CI_SRC=path/to/source/file.c  
PLATFROMIO_CI_SRC=path/to/source/file.ino  
PLATFROMIO_CI_SRC=path/to/source/directory
```

Commands:

```
pip install -U platformio  
platformio ci --board=<ID_1> --board=<ID_2> --board=<ID_N>
```

Build Now

Kickoff build request. Use this testing changes to your build script.
Make sure to save changes first (Save is at the bottom).

Language

Python 2.7 ▾

Databases

- MySQL
- PostgreSQL
- [more ...](#)

Environment Variables

Enter one per line (ex: FOO=bar)

Commands

Enter one command per line (ex: echo foo)

Save

For more details as for PlatformIO build process please look into [*platformio ci*](#) command.

Examples

1. Integration for [USB_Host_Shield_2.0](#) project. The `circle.yml` configuration file:

Environment Variables:

```
PLATFORMIO_CI_SRC=examples/Bluetooth/PS3SPP/PS3SPP.ino
PLATFORMIO_CI_SRC=examples/p12303/p12303_gps/p12303_gps.ino
```

Commands:

```
pip install -U platformio
wget https://github.com/xxxajk/spi4teensy3/archive/master.zip -O /tmp/spi4teensy3.zip
unzip /tmp/spi4teensy3.zip -d /tmp
platformio ci --lib="." --lib="/tmp/spi4teensy3-master" --board=uno --board=teensy31 --
--board=due
```

Build Now

Kickoff build request. Use this testing changes to your build script.
Make sure to save changes first (Save is at the bottom).

Language

Python 2.7

Databases

- MySQL
- PostgreSQL
- [more ...](#)

Environment Variables

```
PLATFORMIO_CI_SRC=examples/Bluetooth/PS3SPP/PS3SPP.ino
PLATFORMIO_CI_SRC=examples/pl2303/pl2303_gps/pl2303_gps.ino
```

Enter one per line (ex: FOO=bar)

Commandsworking directory `/home/ubuntu/src/github.com/valeros/USB_Host_Shield_2.0`

```
pip install --egg
http://sourceforge.net/projects/scons/files/latest/download
pip install
https://github.com/platformio/platformio/archive/develop.zip
wget https://github.com/xxxajk/spi4teensy3/archive/master.zip -O
/tmp/spi4teensy3.zip
unzip /tmp/spi4teensy3.zip -d /tmp
```

Enter one command per line (ex: echo foo)

Save**1.19.4 GitLab**

GitLab is a hosted cloud platform that can help you build, test, deploy, and monitor your code from GitLab repositories.

GitLab CI is enabled by default on new projects, so you can start using its features right away. All you need is `platformio ci` command, a file called `.gitlab-ci.yml` (where you describe how the build should run) placed in the root directory of your git project, and a configured Runner to perform the actual build (Gitlab has some pre-configured public runners so your CI script should work out of the box). Each project comes with a Builds page where you can follow the output of each build, see the commit that introduced it and other useful information such as the time the build started, how long it lasted and the committer's name. The statuses for each build are exposed in the GitLab UI, and you can see whether a build succeeded, failed, got canceled or skipped.

Contents

- *GitLab*
 - *Integration*
 - *Examples*

Integration

Please put `.gitlab-ci.yml` to the root directory of the repository.

```
image: python:2.7

stages:
- test

before_script:
- "pip install -U platformio"

job:
stage: test
script: "platformio ci --board=<ID_1> --board=<ID_2> --board=<ID_N>"
variables: {PLATFORMIO_CI_SRC: "path/to/test/file.c"}
```

For more details as for PlatformIO build process please look into `platformio ci` command.

Examples

1. Integration for ArduinoJson library project. The `.gitlab-ci.yml` configuration file:

```
image: python:2.7

stages:
- test

.job_template: &pio_run
script:
- "platformio ci --lib='.' --board=uno --board=teensy31 --board=nodemcuv2_"
  ↪$PLATFORMIO_CI_EXTRA_ARGS"

before_script:
- "pip install -U platformio"

JsonGeneratorExample:
<<: *pio_run
```

(continues on next page)

(continued from previous page)

```

variables:
  PLATFORMIO_CI_EXTRA_ARGS: "--board=due"
  PLATFORMIO_CI_SRC: examples/JsonGeneratorExample

JsonHttpClient:
<<: *pio_run
variables:
  PLATFORMIO_CI_SRC: examples/JsonHttpClient

JsonParserExample:
<<: *pio_run
variables:
  PLATFORMIO_CI_SRC: examples/JsonParserExample

JsonServer:
<<: *pio_run
variables:
  PLATFORMIO_CI_SRC: examples/JsonServer

JsonUdpBeacon:
<<: *pio_run
variables:
  PLATFORMIO_CI_SRC: examples/JsonUdpBeacon

ProgmemExample:
stage: test
<<: *pio_run
variables:
  PLATFORMIO_CI_SRC: examples/ProgmemExample

StringExample:
stage: test
<<: *pio_run
variables:
  PLATFORMIO_CI_SRC: examples/StringExample

```

1.19.5 Jenkins

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and deploying software.

Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

It can be configured to build project on a range of different *Development Platforms*.

Contents

- *Jenkins*
 - *Integration*

Integration

See step-by-step guide in ThingForward's blog post [Setting up a Jenkins CI engine for embedded projects](#).

1.19.6 Shippable

Shippable is a hosted cloud platform that provides hosted continuous integration, deployment, and testing to GitHub and BitBucket repositories. Shippable's continuous integration service is built using Docker.

Shippable is configured by adding a file named `shippable.yml`, which is a [YAML](#) format text file, to the root directory of the GitHub repository or you can use your Travis CI configuration file `.travis.yml`.

Shippable automatically detects when a commit has been made and pushed to a repository that is using Shippable, and each time this happens, it will try to build the project using `platformio ci` command. This includes commits to all branches, not just to the master branch. Shippable will also build and run pull requests. When that process has completed, it will notify a developer in the way it has been configured to do so — for example, by sending an email containing the build results (showing success or failure), or by posting a message on an IRC channel. It can be configured to build project on a range of different [Development Platforms](#).

Contents

- *Shippable*
 - *Integration*
 - *Examples*

Integration

Please put `shippable.yml` or `.travis.yml` to the root directory of the GitHub repository.

```
language: python
python:
  - "2.7"

env:
  - PLATFORMIO_CI_SRC=path/to/source/file.c
  - PLATFORMIO_CI_SRC=path/to/source/file.ino
  - PLATFORMIO_CI_SRC=path/to/source/directory

install:
  - pip install -U platformio

script:
  - platformio ci --board=<ID_1> --board=<ID_2> --board=<ID_N>
```

For more details as for PlatformIO build process please look into `platformio ci` command.

Examples

1. Integration for [USB_Host_Shield_2.0](#) project. The `shippable.yml` or `.travis.yml` configuration file:

```

language: python
python:
  - "2.7"

env:
  - PLATFORMIO_CI_SRC=examples/Bluetooth/PS3SPP/PS3SPP.ino
  - PLATFORMIO_CI_SRC=examples/pl2303/pl2303_gps/pl2303_gps.ino

install:
  - pip install -U platformio
  - wget https://github.com/xxxajk/spi4teensy3/archive/master.zip -O /tmp/
  ↵spi4teensy3.zip
  - unzip /tmp/spi4teensy3.zip -d /tmp

script:
  - platformio ci --lib="." --lib="/tmp/spi4teensy3-master" --board=uno --
  ↵board=teensy31 --board=due

```

1.19.7 Travis CI



Travis CI officially supports PlatformIO for Embedded Builds.

Travis CI is an open-source hosted, distributed continuous integration service used to build and test projects hosted at GitHub.

Travis CI is configured by adding a file named `.travis.yml`, which is a [YAML](#) format text file, to the root directory of the GitHub repository.

Travis CI automatically detects when a commit has been made and pushed to a repository that is using Travis CI, and each time this happens, it will try to build the project using `platformio ci` command. This includes commits to all branches, not just to the master branch. Travis CI will also build and run pull requests. When that process has completed, it will notify a developer in the way it has been configured to do so — for example, by sending an email containing the build results (showing success or failure), or by posting a message on an IRC channel. It can be configured to build project on a range of different [Development Platforms](#).

Contents

- [Travis CI](#)
 - [Integration](#)
 - [Project as a library](#)
 - [Library dependencies](#)
 - * [Install dependent library using Library Manager](#)
 - * [Manually download dependent library and include in build process via `--lib` option](#)

- *Custom Build Flags*
- *Advanced configuration*
- *Unit Testing*
- *Examples*

Integration

Please make sure to read [Travis CI Getting Started](#) and [general build configuration](#) guides first.

Note: If you are going to use PlatformIO [PIO Unit Testing](#) or [PIO Remote](#) you will need to define `PLATFROMIO_AUTH_TOKEN` environment variable in project settings. See [Defining Variables in Repository Settings](#) guide.

PlatformIO is written in Python and is recommended to be run within [Travis CI Python isolated environment](#):

```
language: python
python:
  - "2.7"

# Cache PlatformIO packages using Travis CI container-based infrastructure
sudo: false
cache:
  directories:
    - "~/.platformio"

env:
  - PLATFROMIO_CI_SRC=path/to/test/file.c
  - PLATFROMIO_CI_SRC=examples/file.ino
  - PLATFROMIO_CI_SRC=path/to/test/directory

install:
  - pip install -U platformio
  - platformio update

script:
  - platformio ci --board=<ID_1> --board=<ID_2> --board=<ID_N>
```

Then perform steps 1, 2 and 4 from <http://docs.travis-ci.com/user/getting-started/>

For more details as for PlatformIO build process please look into [platformio ci](#).

Project as a library

When project is written as a library (where own examples or testing code use it), please use `--lib=". "` option for [platformio ci](#) command

```
script:
  - platformio ci --lib=". " --board=<ID_1> --board=<ID_2> --board=<ID_N>
```

Library dependencies

There 2 options to test source code with dependent libraries:

Install dependent library using Library Manager

```
install:
  - pip install -U platformio

  #
  # Libraries from PlatformIO Library Registry:
  #
  # https://platformio.org/lib/show/1/OneWire
  - platformio lib -g install 1
```

Manually download dependent library and include in build process via `--lib` option

```
install:
  - pip install -U platformio

  # download library to the temporary directory
  - wget https://github.com/PaulStoffregen/OneWire/archive/master.zip -O /tmp/
  ↵onewire_source.zip
  - unzip /tmp/onewire_source.zip -d /tmp/

script:
  - platformio ci --lib="/tmp/OneWire-master" --board=<ID_1> --board=<ID_2> --board=
  ↵<ID_N>
```

Custom Build Flags

PlatformIO allows to specify own build flags using `PLATFORMIO_BUILD_FLAGS` environment

```
env:
  - PLATFORMIO_CI_SRC=path/to/test/file.c PLATFORMIO_BUILD_FLAGS="-D SPECIFIC_
  ↵MACROS_PER_TEST_ENV -I/extra/inc"
  - PLATFORMIO_CI_SRC=examples/file.ino
  - PLATFORMIO_CI_SRC=path/to/test/directory

install:
  - pip install -U platformio
  - export PLATFORMIO_BUILD_FLAGS="-D GLOBAL_MACROS_FOR_ALL_TEST_ENV"
```

For the more details, please follow to [available build flags/options](#).

Advanced configuration

PlatformIO allows to configure multiple build environments for the single source code using [Project Configuration File `platformio.ini`](#).

Instead of `--board` option, please use `platformio ci --project-conf`

```
script:
  - platformio ci --project-conf=/path/to/platformio.ini
```

Unit Testing

See PlatformIO Remote Unit Testing Example.

Examples

1. Custom build flags

```
language: python
python:
  - "2.7"

# Cache PlatformIO packages using Travis CI container-based infrastructure
sudo: false
cache:
  directories:
    - "~/.platformio"

env:
  - PLATFORMIO_CI_SRC=examples/acm/acm_terminal
  - PLATFORMIO_CI_SRC=examples/Bluetooth/WiiIRCamera PLATFORMIO_BUILD_FLAGS="-DWIICAMERA"
  - PLATFORMIO_CI_SRC=examples/ftdi/USBFTDILoopback
  - PLATFORMIO_CI_SRC=examples/Xbox/XBOXUSB
  # - ...

install:
  - pip install -U platformio
  - platformio update

  #
  # Libraries from PlatformIO Library Registry:
  #
  # https://platformio.org/lib/show/416/TinyGPS
  # https://platformio.org/lib/show/417/SPI4Teensy3
  - platformio lib -g install 416 417

script:
  - platformio ci --board=uno --board=teensy31 --board=due --lib=".."
```

- Configuration file: https://github.com/felis/USB_Host_Shield_2.0/blob/master/.travis.yml
- Build History: https://travis-ci.org/felis/USB_Host_Shield_2.0

2. Dependency on external libraries

```
language: python
python:
  - "2.7"

# Cache PlatformIO packages using Travis CI container-based infrastructure
sudo: false
```

(continues on next page)

(continued from previous page)

```
cache:
  directories:
    - "~/.platformio"

env:
  - PLATFROMIO_CI_SRC=examples/backSoon/backSoon.ino
  - PLATFROMIO_CI_SRC=examples/etherNode/etherNode.ino
  # -
  install:
    - pip install -U platformio
    - platformio update

    - wget https://github.com/jcw/jeelib/archive/master.zip -O /tmp/jeelib.zip
    - unzip /tmp/jeelib.zip -d /tmp

    - wget https://github.com/Rodot/Gamebuino/archive/master.zip -O /tmp/gamebuino.
  ↪zip
    - unzip /tmp/gamebuino.zip -d /tmp

script:
  - platformio ci --lib=". " --lib="/tmp/jeelib-master" --lib="/tmp/Gamebuino-master/
  ↪libraries/tinyFAT" --board=uno --board=megaatmega2560
```

- Configuration file: <https://github.com/jcw/ethercard/blob/master/.travis.yml>
 - Build History: <https://travis-ci.org/jcw/ethercard>

3. Dynamic testing of the boards

```
language: python
python:
  - "2.7"

# Cache PlatformIO packages using Travis CI container-based infrastructure
sudo: false
cache:
  directories:
    - "~/.platformio"

env:
  - PLATFORMIO_CI_SRC=examples/TimeArduinoDue PLATFORMIO_CI_EXTRA_ARGS="--board=due"
  - PLATFORMIO_CI_SRC=examples/TimeGPS
  - PLATFORMIO_CI_SRC=examples/TimeNTP
  - PLATFORMIO_CI_SRC=examples/TimeTeensy3 PLATFORMIO_CI_EXTRA_ARGS="--board=teensy31"
  # - ...

install:
  - pip install -U platformio
  - platformio update
  - rm -rf ./linux

#
# Libraries from PlatformIO Library Registry:
#
# https://platformio.org/lib/show/416/TinyGPS
```

(continues on next page)

(continued from previous page)

```
- platformio lib -g install 416 421 422

script:
- platformio ci --lib=". " --board=uno --board=teensy20pp $PLATFORMIO_CI_EXTRA_ARGS
```

- Configuration file: <https://github.com/ivankravets/Time/blob/master/.travis.yml>
- Build History: <https://travis-ci.org/ivankravets/Time>

4. Advanced configuration with extra project options and libraries

```
language: python
python:
- "2.7"

# Cache PlatformIO packages using Travis CI container-based infrastructure
sudo: false
cache:
  directories:
    - "~/.platformio"

env:
- PLATFORMIO_CI_SRC=examples/Boards_Bluetooth/Adafruit_Bluefruit_LE
- PLATFORMIO_CI_SRC=examples/Boards_Bluetooth/Arduino_101_BLE PLATFORMIO_CI_EXTRA_
ARGS="--board=genuino101"
- PLATFORMIO_CI_SRC=examples/Boards_USB_Serial/Blue_Pill_STM32F103C PLATFORMIO_CI_
EXTRA_ARGS="--board=bluepill_f103c8 --project-option='framework=arduino'"
- PLATFORMIO_CI_SRC=examples/Export_Demo/myPlant_ESP8266 PLATFORMIO_CI_EXTRA_ARGS=
"--board=nodemcuv2 --project-option='lib_ignore=Wifi101'"
# - ...

install:
- pip install -U platformio
- platformio update

#
# Libraries from PlatformIO Library Registry:
#
# https://platformio.org/lib/show/44/Time
# https://platformio.org/lib/show/419/SimpleTimer
#
# https://platformio.org/lib/show/17/Adafruit-CC3000
# https://platformio.org/lib/show/28/SPI4Teensy3
# https://platformio.org/lib/show/91/UIPEthernet
# https://platformio.org/lib/show/418/WildFireCore
# https://platformio.org/lib/show/420/WildFire-CC3000
# https://platformio.org/lib/show/65/WiFlyHQ
# https://platformio.org/lib/show/19/Adafruit-DHT
# https://platformio.org/lib/show/299/Wifi101
# https://platformio.org/lib/show/259/BLEPeripheral
# https://platformio.org/lib/show/177/Adafruit_BluefruitLE_nRF51

- platformio lib -g install 17 28 91 418 419 420 65 44 19 299 259 177 https://
github.com/vshymanskyy/BlynkESP8266.git https://github.com/cmaglie/FlashStorage.git_
https://github.com/michael71/Timer5.git

script:
- make travis-build
```

- Configuration file: <https://github.com/blynkkk/blynk-library/blob/master/.travis.yml>
- Build History: <https://travis-ci.org/blynkkk/blynk-library>

1.20 Articles about us

Note: If you've written article about PlatformIO and would like it listed on this page, [please edit this page](#).

Here are recent articles/reviews about PlatformIO:

1.20.1 2018

- Jul 3, 2018 - **Andreas Schmidt** - IoT for web developers: From zero to firmware, Part II
- Jun 22, 2018 - **Andreas Schmidt** - IoT for web developers, Part I
- Jul 4, 2018 - **ThingForward** - Screen-cast: First steps with PlatformIO's Unified Debugger
- Jun 6, 2018 - **Andreas Schmidt** - ESP32, WebThing API and PlatformIO-style projects
- May 21, 2018 - **Dentella Luca** - ESP32, PlatformIO
- Mar 27, 2018 - **Andreas Schmidt** - Building a Web Of Things REST-API on an Arduino MKR1000 with PlatformIO
- Mar 19, 2018 - **ThingForward** - Webinar: Unit Testing for Embedded with PlatformIO and Qt Creator
- Feb 16, 2018 - **Alex Corvis** - DIY Virtual alike NEST Thermostat with Node-RED
- Jan 24, 2018 - **ThingForward** - Embedded and Cloud - Continuous Integration
- Jan 14, 2018 - **IT4nextgen** - 5 Best Development Software(IDE) for Internet of Things (IOT) in 2018
- Jan 13, 2018 - **Rui Marinho** - Quick start guide to flashing ESP8266-based devices with PlatformIO

1.20.2 2017

- Dec 26, 2017 - **Coyt Barringer** - Programming STM32F103 Blue Pill using USB Bootloader and PlatformIO
- Dec 1, 2017 - **ThingForward** - Using Cloud IDEs for Embedded Development: CodeAnywhere
- Nov 13, 2017 - **ThingForward** - Automating Static and Dynamic Code Analysis with PlatformIO
- Oct 18, 2017 - **ThingForward** - Getting Started with SigFox and PlatformIO
- Sep 18, 2017 - **ThingForward** - Embedded Testing with PlatformIO – Part 4: Continuous Integration
- Sep 06, 2017 - **ThingForward** - Embedded Testing with PlatformIO – Part 3: Remoting
- Sep 04, 2017 - **Dror Gluska** - Looking To The IoT Future With PlatformIO And ESP32
- Aug 23, 2017 - Develop ESP32 With PlatformIO IDE
- Aug 08, 2017 - **ThingForward** - Embedded Testing with PlatformIO - Part 2
- Jul 25, 2017 - **ThingForward** - Start Embedded Testing with PlatformIO
- Jun 23, 2017 - **Naresh Krish** - Home Automation Using Wiscore, OpenHab and PlatformIO

- Jun 05, 2017 - **Projects DIY** - Démarrer avec PlatformIO IDE alternatif pour Arduino, ESP8266, ESP32 et autres micro-contrôleurs (Start with PlatformIO alternative IDE for Arduino, ESP8266, ESP32 and other micro-controllers, French)
- Apr 12, 2017 - **Jane Elizabeth** - Let's talk IoT: PlatformIO puts developers back in the driver's seat
- Apr 07, 2017 - **Al Williams** - Hackaday: PlatformIO and Visual Studio take over the world
- Mar 13, 2017 - **Ryan Mulligan** - Continuous testing for Arduino libraries using PlatformIO and Travis CI
- Feb 23, 2017 - **Bastiaan Visee** - Using PlatformIO for your Arduino projects
- Jan 12, 2017 - **Tiest van Gool** - OTA: PlatformIO and ESP8266

1.20.3 2016

- Dec 13, 2016 - **Dr. Patrick Mineault** - Multi-Arduino projects with PlatformIO
- Dec 08, 2016 - **Cuong Tran Viet** - PlatformIO is a solution
- Nov 10, 2016 - **PiGreek** - PlatformIO the new Arduino IDE ?!
- Oct 31, 2016 - **Ricardo Quesada** - Retro Challenge: announcing Commodore Home
- Oct 3, 2016 - **Xose Pérez** - Using the new Bean Loader CLI from PlatformIO
- Sep 20, 2016 - **The Linux Foundation** - 21 Open Source Projects for IoT
- Sep 19, 2016 - **Doc Walker** - How to automatically test build Arduino libraries
- Sep 18, 2016 - **Kadda Sahnine** - LoRaWAN network practice with Objenious, PlatformIO and Node-RED
- Sep 13, 2016 - **Xose Pérez** MQTT LED Matrix Display
- Sep 12, 2016 - **Pedro Minatel** - OTA – Como programar o ESP8266 pelo WiFi no platformIO (OTA programming for ESP8266 via Wi-Fi using PlatformIO, Portuguese)
- Sep 2, 2016 - **Xose Pérez** ESP8266: Optimizing files for SPIFFS with Gulp
- Aug 28, 2016 - **Tom Parker** Using the BBC micro:bit with PlatformIO
- Aug 24, 2016 - **Primal Cortex** Cloud based continuous integration and delivery for IOT using PlatformIO
- Aug 18, 2016 - **Primal Cortex** - Installing PlatformIO on Arch Linux
- Aug 14, 2016 - **Rodrigo Castro** - PlataformIO o comó usar Arduino con ATOM (Spanish)
- Jul 27, 2016 - **Francesco Azzola** - Arduino Alternative IDE: PlatformIO IoT integrated platform
- Jul 26, 2016 - **Embedded Systems Laboratory** - PlatformIO IDE Arduino ESP8266 (Get started with PlatformIO IDE for Arduino board and ESP8266, Thai)
- Jul 15, 2016 - **Jaime** - ESP8266 Mobile Rick Roll Captive Portal
- Jul 5, 2016 - **Ivan Kravets, Ph.D.** - Explore the new development instruments for Arduino with PlatformIO ecosystem
- Jul 5, 2016 - **Belinda** - Monte Bianco Arduino Developer Summit
- Jul 1, 2016 - **Tam Hanna** - Mikrocontroller-Gipfel in den Alpen: Arduino Developer Summit, Tag eins (Microcontroller peaks in the Alps: Arduino Developer Summit, Day One, German)
- Jun 14, 2016 - **Glyn Hudson** - OpenEnergyMonitor Part 2/3: Firmware Continuous Test & Build
- Jun 13, 2016 - **Daniel Eichhorn** - New Weather Station Demo on Github

- Jun 12, 2016 - **Glyn Hudson** - OpenEnergyMonitor Part 1/3: PlatformIO open-source embedded development ecosystem
- Jun 12, 2016 - **Uli Wolf** - Nutzung von PlatformIO im Atom Editor zur Entwicklung von Arduino Code (Use PlatformIO and Atom Editor to develop Arduino code, German)
- Jun 3, 2016 - **Daniel Eichhorn** - ESP8266: Continuous Delivery Pipeline – Push To Production
- May 30, 2016 - **Ron Moerman** - IoT Development with PlatformIO
- May 29, 2016 - **Chris Synan** - Reverse Engineer RF Remote Controller for IoT!
- May 26, 2016 - **Charlie Key** - 7 Best Developer Tools To Build Your NEXT Internet of Things Application
- May 22, 2016 - **Pedro Minatel** - Estação meteorológica com ESP8266 (Weather station with ESP8266, Portuguese)
- May 16, 2016 - **Pedro Minatel** - Controle remoto WiFi com ESP8266 (WiFi remote control using ESP8266, Portuguese)
- May 11, 2016 - **Jo Vandeginste** - Using PlatformIO to compile for Jeelabs' Jeenode Micro
- May 08, 2016 - **Radoslaw Bob** - Touch controlled buzzer (Nodemcu ESP8266)
- May 06, 2016 - **Jean Roux** - The IoT building blocks I use for my home-automation projects
- May 05, 2016 - **Ivan Kravets, Ph.D. / Eclipse Virtual IoT Meetup** - PlatformIO: a cross-platform IoT solution to build them all!
- May 01, 2016 - **Pedro Minatel** - PlatformIO – Uma alternativa ao Arduino IDE (PlatformIO - An alternative to the Arduino IDE, Portuguese)
- Apr 23, 2016 - **Al Williams** - Hackaday: Atomic Arduino (and Other) Development
- Apr 16, 2016 - **Sathittham Sangthong** - [PlatformIO] PlatformIO Arduino IDE (Let's play together with PlatformIO IDE [alternative to Arduino IDE], Thai)
- Apr 15, 2016 - **Daniel Eichhorn** - ESP8266: Offline Debugging with the Platformio Environment
- Apr 11, 2016 - **Matjaz Trcek** - Top 5 Arduino integrated development environments
- Apr 06, 2016 - **Aleks** - PlatformIO ausprobiert (Tried PlatformIO, German)
- Apr 02, 2016 - **Diego Pinto** - Você tem coragem de abandonar a IDE do Arduino? PlatformIO + Atom (Do you dare to leave the Arduino IDE? PlatformIO + Atom, Portuguese)
- Mar 30, 2016 - **Brandon Cannaday** - Getting Started with PlatformIO and ESP8266 NodeMcu
- Mar 29, 2016 - **Pablo Peñalve** - PlatformIO + Geany + Raspberry PI, Spanish
- Mar 24, 2016 - **NAzT** - PlatformIO Arduino Library (PlatformIO and advanced development for Arduino Library, Thai)
- Mar 16, 2016 - **Jakub Skořepa** - Instalace PlatformIO (PlatformIO IDE Installation, Czech)
- Mar 12, 2016 - **Peter Marks** - PlatformIO, the Arduino IDE for programmers
- Mar 12, 2016 - **Richard Arthurs** - Getting Started With PlatformIO
- Mar 07, 2016 - **Joran Jessurun** - Nieuwe wereld met PlatformIO (New world with PlatformIO, Dutch)
- Mar 05, 2016 - **brichacek.net** - PlatformIO – otevřený ekosystém pro vývoj IoT (PlatformIO – an open source ecosystem for IoT development, Czech)
- Mar 04, 2016 - **Ricardo Vega** - Programa tu Arduino desde Atom (Program your Arduino from Atom, Spanish)
- Feb 28, 2016 - **Alex Bloggt** - PlatformIO vorgestellt (Introduction to PlatformIO IDE, German)

- Feb 25, 2016 - **NutDIY** - PlatformIO Blink On Nodemcu Dev Kit V1.0 (Thai)
- Feb 23, 2016 - **Ptarmigan Labs** - ESP8266 Over The Air updating – what are the options?
- Feb 22, 2016 - **Grzegorz Hołdys** - How to Integrate PlatformIO with Netbeans
- Feb 19, 2016 - **Embedds** - Develop easier with PlatformIO ecosystem
- Feb 13, 2016 - **Robert Cudmore** - Programming an arduino with PlatformIO
- Jan 24, 2016 - **Sergey Prilukin** - How to use IntelliJ IDEA to develop and upload software for micro controllers like Arduino
- Jan 16, 2016 - **Dani Eichhorn** - ESP8266 Arduino IDE Alternative: PlatformIO
- Jan 11, 2016 - **David Mills, Ph.D.** - STM NUCLEOF401RE TIMER IO
- Jan 05, 2016 - **Julien Rodrigues** - Internet Of Things: The IDE scandal

1.20.4 2015

- Dec 22, 2015 - **Jan Penninkhof** - Over-the-Air ESP8266 programming using PlatformIO
- Dec 15, 2015 - **stastaka** - PlatformIO (Use a custom board for PlatformIO, Japanese)
- Dec 08, 2015 - **Piotr Król** - Using PlatformIO with TI MSP430 LaunchPads
- Dec 01, 2015 - **Michał Seroczyński** - Push Notification from Arduino Yún with motion sensor
- Dec 01, 2015 - **JetBrains CLion Blog** - C++ Annotated: Fall 2015. Arduino Support in CLion using PlatformIO
- Dec 01, 2015 - **Tateno Yuichi** - ESP8266 CUI (Develop a ESP8266 in CUI, Japanese)
- Nov 29, 2015 - **Keith Hughes** - Using PlatformIO for Embedded Projects
- Nov 22, 2015 - **Michał Seroczyński** - Using PlatformIO to get started with Arduino in CLion IDE
- Nov 09, 2015 - **ÁLvaro García Gómez** - Programar con Arduino “The good way” (Programming with Arduino “The good way”, Spanish)
- Nov 06, 2015 - **nocd5** - PlatformIOembedSTM32 Nucleomruby (Use mruby in the offline build for STM32 Nucleo board with mbed and PlatformIO, Japanese)
- Oct 21, 2015 - **Vittorio Zaccaria** - Using a cheap STM32 Nucleo to teach remote sensor monitoring
- Oct 18, 2015 - **Nico Coetzee** - First Arduino I2C Experience with PlatformIO
- Oct 10, 2015 - **Floyd Hilton** - Programming Arduino with Atom
- Oct 01, 2015 - **Mistan** - Compile and Upload Arduino Sketch with PlatformIO for Raspberry Pi Running Arch Linux
- Sep 30, 2015 - **Jay Wiggins** - PlatformIO Investigation
- Sep 01, 2015 - **Thomas P. Weldon, Ph.D.** - Improvised MBED FRDM-K64F Eclipse/PlatformIO Setup and Software Installation
- Aug 08, 2015 - **Josh Glendenning** - Armstrap Eagle and PlatformIO
- Aug 01, 2015 - **Russell Davis** - PlatformIO on the Raspberry Pi
- Jul 25, 2015 - **DinoTools** - Erste Schritte mit PlatformIO (Getting Started with PlatformIO, German)
- Jul 20, 2015 - **Eli Fatsi** - Arduino Development in Atom Editor
- Jul 14, 2015 - **ElbinarIO** - Programar para Arduino y otros microcontroladores desde la linea de comandos (Program Arguino and other microcontrollers from the command line, Spanish)

- Jul 11, 2015 - **TrojanC** - Learning Arduino GitHub Repository
- Jul 07, 2015 - **Sho Hashimoto** - PlatformIOArduino(Arduino development in PlatformIO, Japanese)
- Jun 02, 2015 - **Alejandro Guirao Rodríguez** - Discovering PlatformIO: The RaspberryPi / Arduino combo kit is a winner option when prototyping an IoT-style project
- May 17, 2015 - **S.S** - Arduino : vim + platformio (Arduino development at the command line: VIM + PlatformIO, Japanese)
- May 11, 2015 - **IT Hare** - From Web Developer to Embedded One: Interview with Ivan Kravets, The Guy Behind PlatformIO. Part II
- May 4, 2015 - **IT Hare** - From Web Developer to Embedded One: Interview with Ivan Kravets, The Guy Behind PlatformIO. Part I
- Apr 17, 2015 - **Michael Ball** - PlatformIO - A Cross-Platform Code Builder and Missing Library Manager
- Mar 23, 2015 - **Atmel** - Cross-board and cross-vendor embedded development with PlatformIO
- Mar 22, 2015 - **Mark VandeWettering** - Discovered a new tool for embedded development: PlatformIO
- Feb 25, 2015 - **Hendrik Putzek** - Use your favourite IDE together with Arduino

1.20.5 2014

- Oct 7, 2014 - **Ivan Kravets, Ph.D.** - Integration of PlatformIO library manager to Arduino and Energia IDEs
- Jun 20, 2014 - **Ivan Kravets, Ph.D.** - Building and debugging Atmel AVR (Arduino-based) project using Eclipse IDE+PlatformIO
- Jun 17, 2014 - **Ivan Kravets, Ph.D.** - How was PlatformIO born or why I love Python World

1.21 Frequently Asked Questions

Note: We have a big database with Frequently Asked Questions in our Community Forums. Please have a look at it.

Contents

- *General*
 - *What is PlatformIO?*
 - *What is .pioenvs directory*
 - *Command completion in Terminal*
 - * *Bash completion*
 - * *ZSH completion*
- *Install Python Interpreter*
- *Convert Arduino file to C++ manually*
- *Program Memory Usage*
- *Troubleshooting*

- *Installation*
 - * *Multiple PIO Cores in a system*
 - * *'platformio' is not recognized as an internal or external command*
 - * *99-platformio-udev.rules*
 - * *ImportError: cannot import name _remove_dead_weakref*
- *Package Manager*
 - * *[Error 5] Access is denied*
 - *Solution 1: Remove folder*
 - *Solution 2: Antivirus*
 - *Solution 3: Run from Terminal*
 - *Solution 4: Manual*
- *Building*
 - * *UnicodeWarning: Unicode equal comparison failed*
 - * *UnicodeDecodeError: Non-ASCII characters found in build environment*
 - * *ARM toolchain: cc1plus: error while loading shared libraries*
 - * *Archlinux: libncurses.so.5: cannot open shared object file*
 - * *Monitoring a serial port breaks upload*

1.21.1 General

What is PlatformIO?

Please refer to *What is PlatformIO?*

What is .pioenvs directory

Please refer to *build_dir*.

Command completion in Terminal

Bash completion

Bash completion support will complete subcommands and parameters. To enable Bash completion for *platformio* subcommands you need to put into your *.bashrc*:

```
eval "$(_PLATFORMIO_COMPLETE=source platformio)"  
eval "$(_PIO_COMPLETE=source pio)"
```

ZSH completion

To enable zsh completion please run these commands:

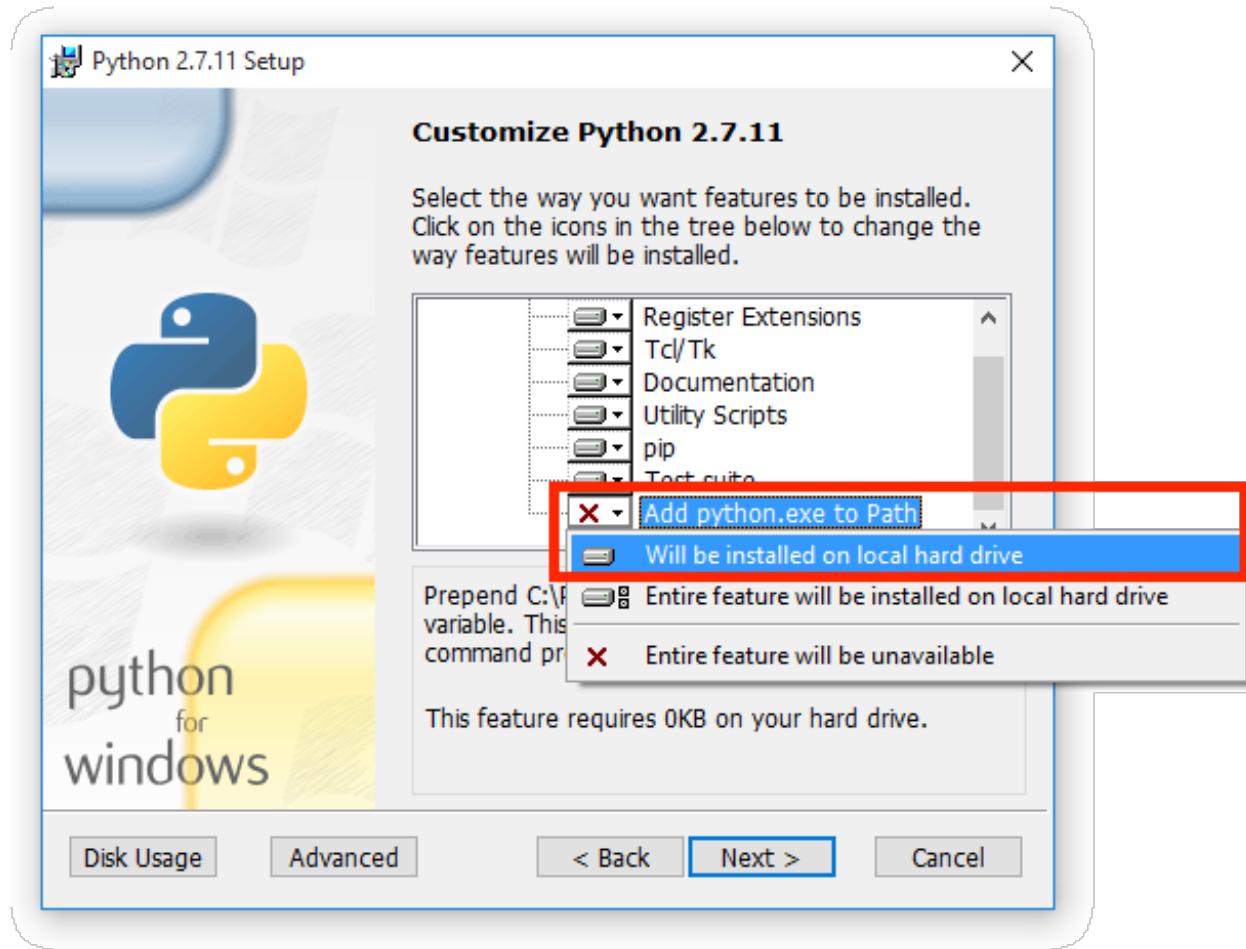
```
autoload bashcompinit && bashcompinit
eval "$(_PLATFORMIO_COMPLETE=source platformio)"
eval "$(_PIO_COMPLETE=source pio)"
```

Note: For permanent command completion you need to place commands above to `~/.bashrc` or `~/.zshrc` file.

1.21.2 Install Python Interpreter

PlatformIO Core is written in [Python](#) that is installed by default on the all popular OS except Windows.

Windows Users, please Download the latest [Python 2.7.x](#) and install it. **DON'T FORGET** to select Add `python.exe` to Path feature on the “Customize” stage, otherwise `python` command will not be available.



1.21.3 Convert Arduino file to C++ manually

Some [Cloud & Desktop IDE](#) doesn't support Arduino files (`*.ino` and `.pde`) because they are not valid C/C++ based source files:

1. Missing includes such as `#include <Arduino.h>`
2. Function declarations are omitted.

In this case, code completion and code linting does not work properly or disabled. To avoid this issue you can manually convert your INO files to CPP.

For example, we have the next Demo.ino file:

```
void setup () {
    someFunction(13);
}

void loop() {
    delay(1000);
}

void someFunction(int num) {
```

Let's convert it to Demo.cpp:

1. Add `#include <Arduino.h>` at the top of the source file
2. Declare each custom function (excluding built-in, such as `setup` and `loop`) before it will be called.

The final Demo.cpp:

```
#include <Arduino.h>

void someFunction(int num);

void setup () {
    someFunction(13);
}

void loop() {
    delay(1000);
}

void someFunction(int num) {
```

1.21.4 Program Memory Usage

PlatformIO calculates firmware/program memory usage based on the next segments:

.text The code segment, also known as a text segment or simply as text, is where a portion of an object file or the corresponding section of the program's virtual address space that contains executable instructions is stored and is generally read-only and fixed size.

.data The .data segment contains any global or static variables which have a pre-defined value and can be modified. The values for these variables are initially stored within the read-only memory (typically within .text) and are copied into the .data segment during the start-up routine of the program. Example,

```
int val = 3;
char string[] = "Hello World";
```

.bss Uninitialized data, is usually adjacent to the data segment. The BSS segment contains all global variables and static variables that are initialized to zero or do not have explicit initialization in source code. For instance, a variable defined as `static int i;` would be contained in the BSS segment.

The rough calculation could be done as:

- PROGRAM (Flash) = .text + .data
- DATA (RAM) = .bss + .data

If you need to print **all memory sections and addresses**, please use `platformio run --verbose` command.

Recommended for reading:

- https://en.wikipedia.org/wiki/Data_segment
- text, data and bss: Code and Data Size Explained

1.21.5 Troubleshooting

Installation

Multiple PIO Cores in a system

Multiple standalone *PlatformIO Core* in a system could lead to a different issues. We highly recommend to keep one instance of PIO Core or use built-in PIO Core in *PlatformIO IDE*:

- *PlatformIO IDE for Atom* - Menu PlatformIO: Settings > PlatformIO IDE > Use built-in PlatformIO Core

Finally, if you have a standalone *PlatformIO Core* in a system, please open system Terminal (not PlatformIO IDE Terminal) and uninstall obsolete PIO Core:

```
pip uninstall platformio

# if you used MacOS "brew"
brew uninstall platformio
```

If you need to have *PlatformIO Core* globally in a system, please [Install Shell Commands](#).

'platformio' is not recognized as an internal or external command

If you use *PlatformIO IDE*, please check in PlatformIO IDE Settings that “Use built-in PIO Core” is enabled.

If you modify system environment variable PATH in your Bash/Fish/ZSH profile, please do not override global PATH. This line `export PATH="/my/custom/path"` is incorrect. Use `export PATH="/my/custom/path":$PATH` instead.

99-platformio-udev.rules

Linux users have to install udev rules for PlatformIO supported boards/devices. The latest version of rules may be found at <https://raw.githubusercontent.com/platformio/platformio-core/develop/scripts/99-platformio-udev.rules>

This file must be placed at `/etc/udev/rules.d/99-platformio-udev.rules` (preferred location) or `/lib/udev/rules.d/99-platformio-udev.rules` (required on some broken systems).

Please open system Terminal and type

```
# Recommended
curl -fsSL https://raw.githubusercontent.com/platformio/platformio-core/develop/
  ↵scripts/99-platformio-udev.rules | sudo tee /etc/udev/rules.d/99-platformio-udev.
  ↵rules

# OR, manually download and copy this file to destination folder
sudo cp 99-platformio-udev.rules /etc/udev/rules.d/99-platformio-udev.rules
```

Restart “udev” management tool:

```
sudo service udev restart

# or

sudo udevadm control --reload-rules
sudo udevadm trigger
```

Ubuntu/Debian users may need to add own “username” to the “dialout” group if they are not “root”, doing this issuing

```
sudo usermod -a -G dialout $USER
sudo usermod -a -G plugdev $USER
```

After this file is installed, physically unplug and reconnect your board.

ImportError: cannot import name _remove_dead_weakref

Windows users can experience this issue when multiple Python interpreters are installed in a system and conflict each other. The easy way to fix this problem is uninstalling all Python interpreters using Windows Programs Manager and installing them manually again.

1. “Windows > Start Menu > Settings > System > Apps & Features”, select Python interpreters and uninstall them.
2. Install the latest Python 2.7 interpreter, see [Install Python Interpreter](#) guide
3. Remove C:\Users\YourUserName\.platformio and C:\.platformio folders if exist (do not forget to replace “YourUserName” with the real user name)
4. Restart [PlatformIO IDE](#).

Package Manager

[Error 5] Access is denied

PlatformIO installs all packages to “`home_dir/packages`” directory. You **MUST HAVE** write access for this folder. Please note that **PlatformIO does not require “sudo”/administrative privileges**.

- [Solution 1: Remove folder](#)
- [Solution 2: Antivirus](#)
- [Solution 3: Run from Terminal](#)
- [Solution 4: Manual](#)

Solution 1: Remove folder

A quick solution is to remove “`home_dir/packages`” folder and repeat installation/building/uploading again.

Solution 2: Antivirus

Some antivirus tools forbid programs to create files in background. PlatformIO Package Manager does all work in background: downloads package, unpacks archive in temporary folder and moves final files to “`home_dir/packages`” folder.

Antivirus tool can block PlatformIO, that is why you see “[Error 5] Access is denied”. Try to **disable it for a while** or add `home_dir` directory to exclusion/white list.

Solution 3: Run from Terminal

As we mentioned in “Solution 2”, antivirus tools can block background file system operations. Other solution is to run *PlatformIO Core* from a system terminal.

1. Open **System Terminal**, on Windows cmd.exe (not *PlatformIO IDE* Terminal)
2. Build project and upload firmware using *PlatformIO Core* which will download and install all dependent packages:

```
# Change directory to PlatformIO Project where is located "platformio.ini"
cd path/to/platformio/project

# Force PlatformIO to install PIO Home dependencies
platformio home

# Force PlatformIO to install toolchains
platformio run --target upload
```

If “platformio” command is not globally available in your environment and you use *PlatformIO IDE*, please use built-in *PlatformIO Core* which is located in:

- Windows: C:\Users\{username}\.platformio\penv\Scripts\platformio Please replace {username} with a real user name
- Unix: ~/.platformio/penv/bin/platformio

Note: You can add `platformio` and `pio` commands to your system environment. See [Install Shell Commands](#).

Solution 4: Manual

If none of solutions above does not work for you, you can download and unpack all packages manually to “`home_dir/packages`”.

Please visit [PlatformIO Package Storage](#) and download a package for your platform. A correct package path is “`home_dir/packages/{package_name}/package.json`”.

Building

UnicodeWarning: Unicode equal comparison failed

Full warning message is “UnicodeWarning: Unicode equal comparison failed to convert both arguments to Unicode - interpreting them as being unequal”.

KNOWN ISSUE. Please move your project to a folder which full path does not contain non-ASCII chars.

UnicodeDecodeError: Non-ASCII characters found in build environment

KNOWN ISSUE. *PlatformIO Core* currently does not support projects which contain non-ASCII characters (codes) in a full path or depend on the libraries which use non-ASCII characters in their names.

TEMPORARY SOLUTION

1. Use *PlatformIO IDE*, it will automatically install *PlatformIO Core* in a root of system disk (%DISK%/.platformio) and avoid an issue when system User contains non-ASCII characters
2. Do not use non-ASCII characters in project folder name or its parent folders.

Also, if you want to place *PlatformIO Core* in own location, see:

- Set `PLATFORMIO_HOME_DIR` environment variable with own path
- Configure custom location per project using `home_dir` option in *Project Configuration File platformio.ini*.

ARM toolchain: cc1plus: error while loading shared libraries

See related answers for error while loading shared libraries.

Archlinux: libncurses.so.5: cannot open shared object file

Answered in issue #291.

Monitoring a serial port breaks upload

Answered in issue #384.

1.22 Release Notes

1.22.1 PlatformIO 3.0

3.6.0 (2018-08-06)

- Program Memory Usage
 - Print human-readable memory usage information after a build and before uploading
 - Print detailed memory usage information with “sections” and “addresses” in verbose mode
 - Check maximum allowed “program” and “data” sizes before uploading/programming (issue #1412)

- PIO Unit Testing:
 - Documented Project Shared Code
 - Force building of project source code using `test_build_project_src` option
 - Fixed missed `UNIT_TEST` macro for unit test components/libraries
- Check package structure after unpacking and raise error when antivirus tool blocks PlatformIO package manager ([issue #1462](#))
- Lock interprocess requests to PlatformIO Package Manager for install/uninstall operations ([issue #1594](#))
- Fixed an issue with [PIO Remote](#) when upload process depends on the source code of a project framework
- Fixed an issue when `srcFilter` field in `library.json` breaks a library build ([issue #1735](#))

3.5.4 (2018-07-03)

- Improved removing of default build flags using `build_unflags` option ([issue #1712](#))
- Export `LIBS`, `LIBPATH`, and `LINKFLAGS` data from project dependent libraries to the global build environment
- Don't export `CPPPATH` data of project dependent libraries to framework's build environment ([issue #1665](#))
- Handle “architectures” data from “library.properties” manifest in `lib_compat_mode = strict`
- Added workaround for Python SemVer package’s issue #61 with caret range and pre-releases
- Replaced conflicted “env” pattern by “sysenv” for “platformio.ini” Dynamic Variables” ([issue #1705](#))
- Removed “date&time” when processing project with `platformio run` command ([issue #1343](#))
- Fixed issue with invalid LD script if path contains space
- Fixed preprocessor for Arduino sketch when function returns certain type ([issue #1683](#))
- Fixed issue when `platformio lib uninstall` removes initial source code ([issue #1023](#))

3.5.3 (2018-06-01)

- PlatformIO Home - interact with PlatformIO ecosystem using modern and cross-platform GUI:
 - “Recent News” block on “Welcome” page
 - Direct import of development platform’s example
- Simplify configuration for [PIO Unit Testing](#): separate main program from a test build process, drop requirement for `#ifdef UNIT_TEST` guard
- Override any option from board manifest in [Project Configuration File “platformio.ini”](#) ([issue #1612](#))
- Configure a custom path to SVD file using `debug_svd_path` option
- Custom project `description` which will be used by PlatformIO Home
- Updated Unity tool to 2.4.3
- Improved support for Black Magic Probe in “uploader” mode
- Renamed “monitor_baud” option to “monitor_speed”
- Fixed issue when a custom `lib_dir` was not handled correctly ([issue #1473](#))
- Fixed issue with useless project rebuilding for case insensitive file systems (Windows)

- Fixed issue with `build_unflags` option when a macro contains value (e.g., `-DNAME=VALUE`)
- Fixed issue which did not allow to override runtime build environment using extra POST script
- Fixed “RuntimeError: maximum recursion depth exceeded” for library manager (issue #1528)

3.5.2 (2018-03-13)

- PlatformIO Home - interact with PlatformIO ecosystem using modern and cross-platform GUI:
 - Multiple themes (Dark & Light)
 - Ability to specify a name for new project
- Control PIO Unified Debugger and its firmware loading mode using `debug_load_mode` option
- Added aliases (off, light, strict) for LDF Compatibility Mode
- Search for a library using PIO Library Registry ID `id:X` (e.g. `pio lib search id:13`)
- Show device system information (MCU, Frequency, RAM, Flash, Debugging tools) in a build log
- Show all available upload protocols before firmware uploading in a build log
- Handle “os.mbed.com” URL as a Mercurial (hg) repository
- Improved support for old mbed libraries without manifest
- Fixed project generator for Qt Creator IDE (issue #1303, issue #1323)
- Mark project source and library directories for CLion IDE (issue #1359, issue #1345, issue #897)
- Fixed issue with duplicated “include” records when generating data for IDE (issue #1301)

3.5.1 (2018-01-18)

- New `test_speed` option to control a communication baudrate/speed between PIO Unit Testing engine and a target device (issue #1273)
- Show full library version in “Library Dependency Graph” including VCS information (issue #1274)
- Configure a custom firmware/program name in build directory (example)
- Renamed `envs_dir` option to `build_dir` in Project Configuration File “`platformio.ini`”
- Refactored code without “arrow” dependency (resolve issue with “ImportError: No module named backports.functools_lru_cache”)
- Improved support of PIO Unified Debugger for Eclipse Oxygen
- Improved a work in off-line mode
- Fixed project generator for CLion and Qt Creator IDE (issue #1299)
- Fixed PIO Unified Debugger for mbed framework
- Fixed library updates when a version is declared in VCS format (not SemVer)

3.5.0 (2017-12-28)

- PlatformIO Home - interact with PlatformIO ecosystem using modern and cross-platform GUI:
 - Library Manager:

- * Search for new libraries in PlatformIO Registry
- * “1-click” library installation, per-project libraries, extra storages
- * List installed libraries in multiple storages
- * List built-in libraries (by frameworks)
- * Updates for installed libraries
- * Multiple examples, trending libraries, and more.
- PlatformIO Projects
- PIO Account
- Development platforms, frameworks and board explorer
- Device Manager: serial, logical, and multicast DNS services
- Integration with Jenkins CI
- New `include` folder for project’s header files ([issue #1107](#))
- Depend on development platform using VCS URL (Git, Mercurial and Subversion) instead of a name in Project Configuration File “`platformio.ini`”. Drop support for `*_stage dev/platform` names (use VCS URL instead).
- Reinstall/redownload package with a new `-f`, `--force` option for `platformio lib install` and `platformio platform install` commands ([issue #778](#))
- Handle missed dependencies and provide a solution based on PlatformIO Library Registry ([issue #781](#))
- New setting `projects_dir` that allows to override a default PIO Home Projects location ([issue #1161](#))
- Library Dependency Finder (LDF):
 - Search for dependencies used in PIO Unit Testing ([issue #953](#))
 - Parse library source file in pair with a header when they have the same name ([issue #1175](#))
 - Handle library dependencies defined as VCS or SemVer in Project Configuration File “`platformio.ini`” ([issue #1155](#))
 - Added option to configure library Compatible Mode using `library.json`
- New options for `platformio device list` command:
 - `--serial` list available serial ports (default)
 - `--logical` list logical devices
 - `--mdns` discover multicast DNS services ([issue #463](#))
- Fixed platforms, packages, and libraries updating behind proxy ([issue #1061](#))
- Fixed missing toolchain include paths for project generator ([issue #1154](#))
- Fixed “Super-Quick (Mac / Linux)” installation in “`get-platformio.py`” script ([issue #1017](#))
- Fixed “`get-platformio.py`” script which hangs on Windows 10 ([issue #1118](#))
- Other bug fixes and performance improvements

3.4.1 (2017-08-02)

- Pre/Post extra scripting for advanced control of PIO Build System ([issue #891](#))
- New `lib_archive` option to control library archiving and linking behavior ([issue #993](#))

- Add “inc” folder automatically to CPPPATH when “src” is available (works for project and library) ([issue #1003](#))
- Use a root of library when filtering source code using `library.json` and `srcFilter` field
- Added `monitor_*` options to white-list for Project Configuration File “`platformio.ini`” ([issue #982](#))
- Do not ask for board ID when initialize project for desktop platform
- Handle broken PIO Core state and create new one
- Fixed an issue with a custom transport for **PIO Unit Testing** when multiple tests are present
- Fixed an issue when can not upload firmware to SAM-BA based board (Due)

3.4.0 (2017-06-26)

- **PIO Unified Debugger**
 - “1-click” solution, zero configuration
 - Support for 100+ embedded boards
 - Multiple architectures and development platforms
 - Windows, MacOS, Linux (+ARMv6-8)
 - Built-in into **PlatformIO IDE for Atom** and **PlatformIO IDE for VScode**
 - Integration with **Eclipse** and **Sublime Text**
- Filter **PIO Unit Testing** tests using a new `test_filter` option in Project Configuration File “`platformio.ini`” or `platformio test -filter` command ([issue #934](#))
- Custom `test_transport` for **PIO Unit Testing Engine**
- Configure Serial Port Monitor in Project Configuration File “`platformio.ini`” ([issue #787](#))
- New `monitor` target which allows to launch Serial Monitor automatically after successful “build” or “upload” operations ([issue #788](#))
- Project generator for **VIM**
- Multi-line support for the different options in Project Configuration File “`platformio.ini`”, such as: `build_flags`, `build_unflags`, etc. ([issue #889](#))
- Handle dynamic `SRC_FILTER` environment variable from `library.json` extra script
- Notify about multiple installations of PIO Core ([issue #961](#))
- Improved auto-detecting of mbed-enabled media disks
- Automatically update Git-submodules for development platforms and libraries that were installed from repository
- Add support for `.*cc` extension ([issue #939](#))
- Handle `env_default` in Project Configuration File “`platformio.ini`” when re-initializing a project ([issue #950](#))
- Use root directory for PIO Home when path contains non-ascii characters ([issue #951](#), [issue #952](#))
- Don’t warn about known `boards_dir` option ([pull #949](#))
- Escape non-valid file name characters when installing a new package (library) ([issue #985](#))
- Fixed infinite dependency installing when repository consists of multiple libraries ([issue #935](#))
- Fixed linter error “unity.h does not exist” for Unit Testing ([issue #947](#))

- Fixed issue when Library Dependency Finder (LDF) does not handle custom `src_dir` ([issue #942](#))
- Fixed cloning a package (library) from a private Git repository with custom user name and SSH port ([issue #925](#))

3.3.1 (2017-05-27)

- Hotfix for recently updated Python Requests package (2.16.0)

3.3.0 (2017-03-27)

- PlatformIO Library Registry statistics with new `pio lib stats` command
 - Recently updated and added libraries
 - Recent and popular keywords
 - Featured libraries (today, week, month)
- List built-in libraries based on development platforms with a new `pio lib builtin` command
- Show detailed info about a library using `pio lib show` command ([issue #430](#))
- List supported frameworks, SDKs with a new `pio platform frameworks` command
- Visual Studio Code extension for PlatformIO ([issue #619](#))
- Added new options `--no-reset`, `--monitor-rts` and `--monitor-dtr` to `pio test` command (allows to avoid automatic board's auto-reset when gathering test results)
- Added support for templated methods in `*.ino` to `*.cpp` converter ([pull #858](#))
- Package version as "Repository URL" in manifest of development version ("version": "https://github.com/user/repo.git")
- Produce less noisy output when `-s`/`--silent` options are used for `platformio init` and `platformio run` commands ([issue #850](#))
- Use C++11 by default for CLion IDE based projects ([pull #873](#))
- Escape project path when Glob matching is used
- Do not overwrite project configuration variables when system environment variables are set
- Handle dependencies when installing non-registry package/library (VCS, archive, local folder) ([issue #913](#))
- Fixed package installing with VCS branch for Python 2.7.3 ([issue #885](#))

3.2.1 (2016-12-07)

- Changed default LDF Mode from `chain+` to `chain`

3.2.0 (2016-12-07)

- **PIO Remote™. Your devices are always with you!**
 - Over-The-Air (OTA) Device Manager
 - OTA Serial Port Monitor
 - OTA Firmware Updates

- Continuous Deployment
- Continuous Delivery
- Integration with Cloud IDEs
 - Cloud9
 - Codeanywhere
 - Eclipse Che
- PIO Account and PLATFORMIO_AUTH_TOKEN environment variable for CI systems (issue #808, issue #467)
- Inject system environment variables to configuration settings in Project Configuration File “platformio.ini” (issue #792)
- Custom boards per project with boards_dir option in Project Configuration File “platformio.ini” (issue #515)
- Unix shell-style wildcards for upload_port (issue #839)
- Refactored Library Dependency Finder (LDF) C/C++ Preprocessor for conditional syntax (#ifdef, #if, #else, #elif, #define, etc.) (issue #837)
- Added new LDF Modes: chain+ and deep+ and set chain+ as default
- Added global lib_extra_dirs option to [platformio] section for Project Configuration File “platformio.ini” (issue #842)
- Enabled caching by default for API requests and Library Manager (see enable_cache setting)
- Native integration with VIM/Neovim using neomake-platformio plugin
- Changed a default exit combination for Device Monitor from Ctrl+] to Ctrl+C
- Improved detecting of ARM mbed media disk for uploading
- Improved Project Generator for CLion IDE when source folder contains nested items
- Improved handling of library dependencies specified in library.json manifest (issue #814)
- Improved Library Dependency Finder (LDF) for circular dependencies
- Show vendor version of a package for platformio platform show command (issue #838)
- Fixed unable to include SSH user in lib_deps repository url (issue #830)
- Fixed merging of “.gitignore” files when re-initialize project (issue #848)
- Fixed issue with PATH auto-configuring for upload tools
- Fixed 99-platformio-udev.rules checker for Linux OS

3.1.0 (2016-09-19)

- New! Dynamic variables/templates for Project Configuration File “platformio.ini” (issue #705)
- Summary about processed environments (issue #777)
- Implemented LocalCache system for API and improved a work in off-line mode
- Improved Project Generator when custom --project-option is passed to platformio init command
- Deprecated lib_force option, please use lib_deps instead
- Return valid exit code from platformio test command

- Fixed Project Generator for CLion IDE using Windows OS (issue #785)
- Fixed SSL Server-Name-Indication for Python < 2.7.9 (issue #774)

3.0.1 (2016-09-08)

- Disabled temporary SSL for PlatformIO services (issue #772)

3.0.0 (2016-09-07)

- PlatformIO Plus
 - Local and Embedded Unit Testing (issue #408, issue #519)
- Decentralized Development Platforms
 - Development platform manifest “platform.json” and open source development platforms
 - Semantic Versioning for platform commands, development platforms and dependent packages
 - Custom package repositories
 - External embedded board configuration files, isolated build scripts (issue #479)
 - Embedded Board compatibility with more than one development platform (issue #456)
- Library Manager 3.0
 - Project dependencies per build environment using lib_deps option (issue #413)
 - Semantic Versioning for library commands and dependencies (issue #410)
 - Multiple library storages: Project’s Local, PlatformIO’s Global or Custom (issue #475)
 - Install library by name (issue #414)
 - Depend on a library using VCS URL (GitHub, Git, ARM mbed code registry, Hg, SVN) (issue #498)
 - Strict search for library dependencies (issue #588)
 - Allowed library.json to specify sources other than PlatformIO’s Repository (issue #461)
 - Search libraries by headers/includes with platformio lib search --header option
- New Intelligent Library Build System
 - Library Dependency Finder that interprets C/C++ Preprocessor conditional macros with deep search behavior
 - Check library compatibility with project environment before building (issue #415)
 - Control Library Dependency Finder for compatibility using lib_compat_mode option
 - Custom library storages/directories with lib_extra_dirs option (issue #537)
 - Handle extra build flags, source filters and build script from library.json (issue #289)
 - Allowed to disable library archiving (*.ar) (issue #719)
 - Show detailed build information about dependent libraries (issue #617)
 - Support for the 3rd party manifests (Arduino IDE “library.properties” and ARM mbed “module.json”)
- Removed enable_prompts setting. Now, all PlatformIO CLI is non-blocking!
- Switched to SSL PlatformIO API

- Renamed `platformio serialports` command to `platformio device`
- Build System: Attach custom Before/Pre and After/Post actions for targets ([issue #542](#))
- Allowed passing custom project configuration options to `platformio ci` and `platformio init` commands using `-O`, `--project-option`.
- Print human-readable information when processing environments without `-v`, `--verbose` option ([issue #721](#))
- Improved INO to CPP converter ([issue #659](#), [issue #765](#))
- Added `license` field to `library.json` ([issue #522](#))
- Warn about unknown options in project configuration file `platformio.ini` ([issue #740](#))
- Fixed wrong line number for INO file when `#warning` directive is used ([issue #742](#))
- Stopped supporting Python 2.6

1.22.2 PlatformIO 2.0

2.11.2 (2016-08-02)

- Improved support for [Microchip PIC32](#) development platform and ChipKIT boards ([issue #438](#))
- Added support for Pinoccio Scout board ([issue #52](#))
- Added support for [Teensy USB Features](#) (HID, SERIAL_HID, DISK, DISK_SDFLASH, MIDI, etc.) ([issue #722](#))
- Switched to built-in GCC LwIP library for Espressif development platform
- Added support for local `--echo` for Serial Port Monitor ([issue #733](#))
- Updated `udev` rules for the new STM32F407DISCOVERY boards ([issue #731](#))
- Implemented firmware merging with base firmware for Nordic nRF51 development platform ([issue #500](#), [issue #533](#))
- Fixed Project Generator for ESP8266 and ARM mbed based projects (resolves incorrect linter errors)
- Fixed broken LD Script for Element14 chipKIT Pi board ([issue #725](#))
- Fixed firmware uploading to Atmel SAMD21-XPRO board using ARM mbed framework ([issue #732](#))

2.11.1 (2016-07-12)

- Added support for Arduino M0, M0 Pro and Tian boards ([issue #472](#))
- Added support for Microchip chipKIT Lenny board
- Updated Microchip PIC32 Arduino framework to v1.2.1
- Documented [uploading of EEPROM data](#) (from EEMEM directive)
- Added `Rebuild C/C++ Project Index` target to CLion and Eclipse IDEs
- Improved project generator for [CLion IDE](#)
- Added `udev` rules for OpenOCD CMSIS-DAP adapters ([issue #718](#))
- Auto-remove project cache when PlatformIO is upgraded
- Keep user changes for `.gitignore` file when re-generate/update project data

- Ignore [platformio] section from custom project configuration file when `platformio ci --project-conf` command is used
- Fixed missed `--boot` flag for the firmware uploader for ATSAM3X8E Cortex-M3 MCU based boards (Arduino Due, etc) ([issue #710](#))
- Fixed missing trailing \ for the source files list when generate project for Qt Creator IDE ([issue #711](#))
- Split source files to `HEADERS` and `SOURCES` when generate project for Qt Creator IDE ([issue #713](#))

2.11.0 (2016-06-28)

- New ESP8266-based boards: Generic ESP8285 Module, Phoenix 1.0 & 2.0, WifInfo
- Added support for Arduino M0 Pro board ([issue #472](#))
- Added support for Arduino MKR1000 board ([issue #620](#))
- Added support for Adafruit Feather M0, SparkFun SAMD21 and SparkFun SAMD21 Mini Breakout boards ([issue #520](#))
- Updated Arduino ESP8266 core for Espressif platform to 2.3.0
- Better removing unnecessary flags using `build_unflags` option ([issue #698](#))
- Fixed issue with `platformio init --ide` command for Python 2.6

2.10.3 (2016-06-15)

- Fixed issue with `platformio init --ide` command

2.10.2 (2016-06-15)

- Added support for ST Nucleo L031K6 board to ARM mbed framework
- Process `build_unflags` option for ARM mbed framework
- Updated Intel ARC32 Arduino framework to v1.0.6 ([issue #695](#))
- Improved a check of program size before uploading to the board
- Fixed issue with ARM mbed framework `-u __printf_float` and `-u __scanf_float` when parsing `$LINKFLAGS`
- Fixed issue with ARM mbed framework and extra includes for the custom boards, such as Seeeduino Arch Pro

2.10.1 (2016-06-13)

- Re-submit a package to PyPI

2.10.0 (2016-06-13)

- Added support for emonPi, the OpenEnergyMonitor system ([issue #687](#))
- Added support for SPL framework for STM32F0 boards ([issue #683](#))
- Added support for Arduboy DevKit, the game system the size of a credit card
- Updated ARM mbed framework package to v121

- Check program size before uploading to the board ([issue #689](#))
- Improved firmware uploading to Arduino Leonardo based boards ([issue #691](#))
- Fixed issue with `-L relative/path` when parsing `build_flags` ([issue #688](#))

2.9.4 (2016-06-04)

- Show `udev` warning only for the Linux OS while uploading firmware

2.9.3 (2016-06-03)

- Added support for [Arduboy](#), the game system the size of a credit card
- Updated [99-platformio-udev.rules](#) for Linux OS
- Refactored firmware uploading to the embedded boards with SAM-BA bootloader

2.9.2 (2016-06-02)

- Simplified [Continuous Integration with AppVeyor](#) ([issue #671](#))
- Automatically add source directory to `CPPPATH` of Build System
- Added support for Silicon Labs SLSTK3401A (Pearl Gecko) and MultiTech mDot F411 ARM mbed based boards
- Added support for MightyCore ATmega8535 board ([issue #585](#))
- Added `stlink` as the default uploader for STM32 Discovery boards ([issue #665](#))
- Use HTTP mirror for Package Manager in a case with SSL errors ([issue #645](#))
- Improved firmware uploading to Arduino Leonardo/Due based boards
- Fixed bug with `env_default` when `pio run -e` is used
- Fixed issue with `src_filter` option for Windows OS ([issue #652](#))
- Fixed configuration data for TI LaunchPads based on msp430fr4133 and msp430fr6989 MCUs ([issue #676](#))
- Fixed issue with ARM mbed framework and multiple definition errors on FRDM-KL46Z board ([issue #641](#))
- Fixed issue with ARM mbed framework when abstract class breaks compile for LPC1768 ([issue #666](#))

2.9.1 (2016-04-30)

- Handle prototype pointers while converting `*.ino` to `.cpp` ([issue #639](#))

2.9.0 (2016-04-28)

- Project generator for [CodeBlocks IDE](#) ([issue #600](#))
- New [Lattice iCE40 FPGA](#) development platform with support for Lattice iCEstick FPGA Evaluation Kit and BQ IceZUM Alhambra FPGA ([issue #480](#))
- New [Intel ARC 32-bit](#) development platform with support for Arduino/Genuino 101 board ([issue #535](#))
- New [Microchip PIC32](#) development platform with support for 20+ different PIC32 based boards ([issue #438](#))

- New RTOS and build Framework named [Simba](#) (issue #412)
- New boards for [ARM mbed](#) framework: ST Nucleo F410RB, ST Nucleo L073RZ and BBC micro:bit
- Added support for Arduino.Org boards: Arduino Leonardo ETH, Arduino Yun Mini, Arduino Industrial 101 and Linino One (issue #472)
- Added support for Generic ATTiny boards: ATTiny13, ATTiny24, ATTiny25, ATTiny45 and ATTiny85 (issue #636)
- Added support for MightyCore boards: ATmega1284, ATmega644, ATmega324, ATmega164, ATmega32, ATmega16 and ATmega8535 (issue #585)
- Added support for [TI MSP430](#) boards: TI LaunchPad w/ msp430fr4133 and TI LaunchPad w/ msp430fr6989
- Updated Arduino core for Espressif platform to 2.2.0 (issue #627)
- Updated native SDK for ESP8266 to 1.5 (issue #366)
- PlatformIO Library Registry in JSON format! Implemented --json-output and --page options for [platformio lib search](#) command (issue #604)
- Allowed to specify default environments `env_default` which should be processed by default with `platformio run` command (issue #576)
- Allowed to unflag(remove) base/initial flags using `build_unflags` option (issue #559)
- Allowed multiple VID/PID pairs when detecting serial ports (issue #632)
- Automatically add `-DUSB_MANUFACTURER` with vendor's name (issue #631)
- Automatically reboot Teensy board after upload when Teensy Loader GUI is used (issue #609)
- Refactored source code converter from `*.ino` to `*.cpp` (issue #610)
- Forced `-std=gnu++11` for Atmel SAM development platform (issue #601)
- Don't check OS type for ARM mbed-enabled boards and ST STM32 development platform before uploading to disk (issue #596)
- Fixed broken compilation for Atmel SAMD based boards except Arduino Due (issue #598)
- Fixed firmware uploading using serial port with spaces in the path
- Fixed cache system when project's root directory is used as `src_dir` (issue #635)

2.8.6 (2016-03-22)

- Launched [PlatformIO Community Forums](#) (issue #530)
- Added support for ARM mbed-enabled board Seed Arch Max (STM32F407VET6) (issue #572)
- Improved DNS lookup for PlatformIO API
- Updated Arduino Wiring-based framework to the latest version for Atmel AVR/SAM development platforms
- Updated "Teensy Loader CLI" and fixed uploading of large .hex files (issue #568)
- Updated the support for Sanguino Boards (issue #586)
- Better handling of used boards when re-initialize/update project
- Improved support for non-Unicode user profiles for Windows OS
- Disabled progress bar for download operations when prompts are disabled
- Fixed multiple definition errors for ST STM32 development platform and ARM mbed framework (issue #571)

- Fixed invalid board parameters (reset method and baudrate) for a few ESP8266 based boards
- Fixed “KeyError: ‘content-length’” in PlatformIO Download Manager (issue #591)

2.8.5 (2016-03-07)

- Project generator for [NetBeans IDE](#) (issue #541)
- Created package for Homebrew Mac OS X Package Manager: `brew install platformio` (issue #395)
- Updated Arduino core for Espressif platform to 2.1.0 (issue #544)
- Added support for the ESP8266 ESP-07 board to Espressif (issue #527)
- Improved handling of String-based `CPPDEFINES` passed to `extra build_flags` (issue #526)
- Generate appropriate project for CLion IDE and CVS (issue #523)
- Use `src_dir` directory from [Project Configuration File](#) `platformio.ini` when initializing project otherwise create base `src` directory (issue #536)
- Fixed issue with incorrect handling of user’s build flags where the base flags were passed after user’s flags to GCC compiler (issue #528)
- Fixed issue with Project Generator when optional build flags were passed using system environment variables: `PLATFORMIO_BUILD_FLAGS` or `PLATFORMIO_BUILD_SRC_FLAGS`
- Fixed invalid detecting of compiler type (issue #550)
- Fixed issue with updating package which was deleted manually by user (issue #555)
- Fixed incorrect parsing of GCC `-include` flag (issue #552)

2.8.4 (2016-02-17)

- Added support for the new ESP8266-based boards (ESPDuino, ESP-WROOM-02, ESPRESSO Lite 1.0 & 2.0, SparkFun ESP8266 Thing Dev, ThaiEasyElec ESPino) to [Espressif](#) development platform
- Added `board_f_flash` option to [Project Configuration File](#) `platformio.ini` which allows to specify [custom flash chip frequency](#) for Espressif development platform (issue #501)
- Added `board_flash_mode` option to [Project Configuration File](#) `platformio.ini` which allows to specify [custom flash chip mode](#) for Espressif development platform
- Handle new environment variables `PLATFORMIO_UPLOAD_PORT` and `PLATFORMIO_UPLOAD_FLAGS` (issue #518)
- Fixed issue with `CPPDEFINES` which contain space and break PlatformIO IDE Linter ([IDE](#) issue #34)
- Fixed unable to link C++ standard library to Espressif platform build (issue #503)
- Fixed issue with pointer (`char* myfunc()`) while converting from `*.ino` to `*.cpp` (issue #506)

2.8.3 (2016-02-02)

- Better integration of PlatformIO Builder with PlatformIO IDE Linter
- Fixed issue with removing temporary file while converting `*.ino` to `*.cpp`
- Fixed missing dependency (mbed framework) for Atmel SAM development platform (issue #487)

2.8.2 (2016-01-29)

- Corrected RAM size for NXP LPC1768 based boards ([issue #484](#))
- Exclude only `test` and `tests` folders from build process
- Reverted `-Wl,-whole-archive` hook for ST STM32 and mbed

2.8.1 (2016-01-29)

- Fixed a bug with Project Initialization in PlatformIO IDE

2.8.0 (2016-01-29)

- PlatformIO IDE for Atom ([issue #470](#))
- Added `pio` command line alias for `platformio` command ([issue #447](#))
- Added SPL-Framework support for Nucleo F401RE board ([issue #453](#))
- Added `upload_resetmethod` option to [Project Configuration File platformio.ini](#) which allows to specify custom upload reset method for Espressif development platform ([issue #444](#))
- Allowed to force output of color ANSI-codes or to disable progress bar even if the output is a pipe (not a `tty`) using [Environment variables](#) ([issue #465](#))
- Set 1Mb SPIFFS for Espressif boards by default ([issue #458](#))
- Exclude `test*` folder by default from build process
- Generate project for IDEs with information about installed libraries
- Fixed builder for mbed framework and ST STM32 platform

2.7.1 (2016-01-06)

- Initial support for Arduino Zero board ([issue #356](#))
- Added support for completions to Atom text editor using `.clang_complete`
- Generate default targets for [supported IDE](#) (CLion, Eclipse IDE, Emacs, Sublime Text, VIM): Build, Clean, Upload, Upload SPIFFS image, Upload using Programmer, Update installed platforms and libraries ([issue #427](#))
- Updated Teensy Arduino Framework to 1.27 ([issue #434](#))
- Fixed uploading of EEPROM data using `uploaddeep` target for Atmel AVR development platform
- Fixed project generator for CLion IDE ([issue #422](#))
- Fixed package `shasum` validation on Mac OS X 10.11.2 ([issue #429](#))
- Fixed `CMakeLists.txt add_executable` has only one source file ([issue #421](#))

2.7.0 (2015-12-30)

Happy New Year!

- Moved SCons to PlatformIO packages. PlatformIO does not require SCons to be installed in your system. Significantly simplified installation process of PlatformIO. `pip install platformio` rocks!
- Implemented uploading files to file system of ESP8266 SPIFFS (including OTA) ([issue #382](#))

- Added support for the new Adafruit boards Bluefruit Micro and Feather ([issue #403](#))
- Added support for RFDuino ([issue #319](#))
- Project generator for [Emacs](#) text editor ([pull #404](#))
- Updated Arduino framework for Atmel AVR development platform to 1.6.7
- Documented [firmware uploading for Atmel AVR development platform using Programmers: AVR ISP, AVRISP mkII, USBtinyISP, USBasp, Parallel Programmer and Arduino as ISP](#)
- Fixed issue with current Python interpreter for Python-based tools ([issue #417](#))

2.6.3 (2015-12-21)

- Restored support for Espressif ESP8266 ESP-01 1MB board (ready for OTA)
- Fixed invalid ROM size for ESP8266-based boards ([issue #396](#))

2.6.2 (2015-12-21)

- Removed `SCons` from requirements list. PlatformIO will try to install it automatically, otherwise users need to install it manually
- Fixed `ChunkedEncodingError` when SF connection is broken ([issue #356](#))

2.6.1 (2015-12-18)

- Added support for the new ESP8266-based boards (SparkFun ESP8266 Thing, NodeMCU 0.9 & 1.0, Olimex MOD-WIFI-ESP8266(-DEV), Adafruit HUZZAH ESP8266, ESPino, SweetPea ESP-210, WeMos D1, WeMos D1 mini) to [Espressif](#) development platform
- Created public [platformio-pkg-ldscripts](#) repository for LD scripts. Moved common configuration for ESP8266 MCU to `esp8266.flash.common.ld` ([issue #379](#))
- Improved documentation for [Espressif](#) development platform: OTA update, custom Flash Size, Upload Speed and CPU frequency
- Fixed reset method for Espressif NodeMCU (ESP-12E Module) ([issue #380](#))
- Fixed issue with code builder when build path contains spaces ([issue #387](#))
- Fixed project generator for Eclipse IDE and “duplicate path entries found in project path” ([issue #383](#))

2.6.0 (2015-12-15)

- Install only required packages depending on build environment ([issue #308](#))
- Added support for Raspberry Pi [WiringPi](#) framework ([issue #372](#))
- Implemented Over The Air (OTA) upgrades for [Espressif](#) development platform. ([issue #365](#))
- Updated [CMSIS](#) framework and added CMSIS support for Nucleo F401RE board ([issue #373](#))
- Added support for Espressif ESP8266 ESP-01-1MB board (ready for OTA)
- Handle `upload_flags` option in `platformio.ini` ([issue #368](#))
- Improved PlatformIO installation on the Mac OS X El Capitan

2.5.0 (2015-12-08)

- Improved code builder for parallel builds (up to 4 times faster than before)
- Generate `.travis.yml` CI and `.gitignore` files for embedded projects by default ([issue #354](#))
- Removed prompt with “auto-uploading” from `platformio init` command and added `--enable-auto-uploading` option ([issue #352](#))
- Fixed incorrect behaviour of `platformio serialports monitor` in pair with PySerial 3.0

2.4.1 (2015-12-01)

- Restored `PLATFORMIO` macros with the current version

2.4.0 (2015-12-01)

- Added support for the new boards: Atmel ATSAMR21-XPRO, Atmel SAML21-XPRO-B, Atmel SAMD21-XPRO, ST 32F469IDISCOVERY, ST 32L476GDISCOVERY, ST Nucleo F031K6, ST Nucleo F042K6, ST Nucleo F303K8 and ST Nucleo L476RG
- Updated Arduino core for Espressif platform to 2.0.0 ([issue #345](#))
- Added to FAQ explanation of Can not compile a library that compiles without issue with Arduino IDE ([issue #331](#))
- Fixed ESP-12E flash size ([pull #333](#))
- Fixed configuration for LowPowerLab MoteinoMEGA board ([issue #335](#))
- Fixed “LockFailed: failed to create appstate.json.lock” error for Windows
- Fixed relative include path for preprocessor using `build_flags` ([issue #271](#))

2.3.5 (2015-11-18)

- Added support of `libOpenCM3` framework for Nucleo F103RB board ([issue #309](#))
- Added support for Espressif ESP8266 ESP-12E board (NodeMCU) ([issue #310](#))
- Added support for pySerial 3.0 ([issue #307](#))
- Updated Arduino AVR/SAM frameworks to 1.6.6 ([issue #321](#))
- Upload firmware using external programmer via `platformio run --target program target` ([issue #311](#))
- Fixed handling of upload port when `board` option is not specified in `platformio.ini` ([issue #313](#))
- Fixed firmware uploading for `nordicrf51` development platform ([issue #316](#))
- Fixed installation on Mac OS X El Capitan ([issue #312](#))
- Fixed project generator for CLion IDE under Windows OS with invalid path to executable ([issue #326](#))
- Fixed empty list with serial ports on Mac OS X ([issue #294](#))
- Fixed compilation error `TWI_Disable` not declared for Arduino Due board ([issue #329](#))

2.3.4 (2015-10-13)

- Full support of CLion IDE including code auto-completion (issue #132)
- PlatformIO command completion in Terminal for bash and zsh
- Added support for ubIQio Ardhat board (pull #302)
- Install SCons automatically and avoid error: option --single-version-externally-managed not recognized (issue #279)
- Use Teensy CLI Loader for upload of .hex files on Mac OS X (issue #306)
- Fixed missing framework-mbed package for teensy platform (issue #305)

2.3.3 (2015-10-02)

- Added support for LightBlue Bean board (pull #292)
- Added support for ST Nucleo F446RE board (pull #293)
- Fixed broken lock file for “appstate” storage (issue #288)
- Fixed ESP8266 compile errors about RAM size when adding 1 library (issue #296)

2.3.2 (2015-09-10)

- Allowed to use ST-Link uploader for mbed-based projects
- Explained how to use lib directory from the PlatformIO based project in `readme.txt` which will be automatically generated using `platformio init` command (issue #273)
- Found solution for “pip/scons error: option –single-version-externally-managed not recognized” when install PlatformIO using pip package manager (issue #279)
- Fixed firmware uploading to Arduino Leonardo board using Mac OS (issue #287)
- Fixed `SConsNotInstalled` error for Linux Debian-based distributives

2.3.1 (2015-09-06)

- Fixed critical issue when `platformio init -ide` command hangs PlatformIO (issue #283)

2.3.0 (2015-09-05)

- Added native, linux_arm, linux_i686, linux_x86_64, windows_x86 development platforms (issue #263)
- Added PlatformIO Demo page to documentation
- Simplified installation process of PlatformIO (issue #274)
- Significantly improved Project Generator which allows to integrate with the most popular IDE
- Added short -h help option for PlatformIO and sub-commands
- Updated mbed framework
- Updated tool-teensy package for Teensy platform (issue #268)
- Added FAQ answer when Program “platformio” not found in PATH (issue #272)

- Generate “readme.txt” for project “lib” directory (issue #273)
- Use toolchain’s includes pattern `include*` for Project Generator (issue #277)
- Added support for Adafruit Gemma board to `atmelavr` platform (pull #256)
- Fixed includes list for Windows OS when generating project for `Eclipse IDE` (issue #270)
- Fixed `AttributeError: 'module' object has no attribute 'packages'` (issue #252)

2.2.2 (2015-07-30)

- Integration with `Atom IDE`
- Support for off-line/unpublished/private libraries (issue #260)
- Disable project auto-clean while building/uploading firmware using `platformio run --disable-auto-clean` option (issue #255)
- Show internal errors from “Miniterm” using `platformio serialports monitor` command (issue #257)
- Fixed `platformio serialports monitor -help` information with HEX char for hotkeys (issue #253)
- Handle “`OSError: [Errno 13] Permission denied`” for PlatformIO installer script (issue #254)

2.2.1 (2015-07-17)

- Project generator for `CLion IDE` (issue #132)
- Updated `tool-bossac` package to 1.5 version for `atmelsam` platform (issue #251)
- Updated `sdk-esp8266` package for `espressif` platform
- Fixed incorrect arguments handling for `platformio serialports monitor` command (issue #248)

2.2.0 (2015-07-01)

- Allowed to exclude/include source files from build process using `src_filter` (issue #240)
- Launch own extra script before firmware building/uploading processes (issue #239)
- Specify own path to the linker script (`ld`) using `build_flags` option (issue #233)
- Specify library compatibility with the all platforms/frameworks using `*` symbol in `library.json`
- Added support for new embedded boards: *ST 32L0538DISCOVERY* and *Delta DFCM-NNN40* to `Framework mbed`
- Updated packages for `Framework Arduino` (AVR, SAM, Espressif and Teensy cores, `Framework mbed`, `Espressif ESP8266 SDK` (issue #246))
- Fixed `stk500v2_command(): command failed` (issue #238)
- Fixed IDE project generator when board is specified (issue #242)
- Fixed relative path for includes when generating project for IDE (issue #243)
- Fixed `ESP8266 native SDK exception` (issue #245)

2.1.2 (2015-06-21)

- Fixed broken link to SCons installer

2.1.1 (2015-06-09)

- Automatically detect upload port using VID:PID board settings ([issue #231](#))
- Improved detection of build changes
- Avoided `LibInstallDependencyError` when more than 1 library is found ([issue #229](#))

2.1.0 (2015-06-03)

- Added Silicon Labs EFM32 `siliconlabsefm32` development platform ([issue #226](#))
- Integrate PlatformIO with [Circle CI](#) and [Shippable CI](#)
- Described in documentation how to [create/register own board](#) for PlatformIO
- Disabled “nano.specs” for ARM-based platforms ([issue #219](#))
- Fixed “ConnectionError” when PlatformIO SF Storage is off-line
- Fixed resolving of C/C++ std libs by Eclipse IDE ([issue #220](#))
- Fixed firmware uploading using USB programmer (USBasp) for `atmelavr` platform ([issue #221](#))

2.0.2 (2015-05-27)

- Fixed libraries order for “Library Dependency Finder” under Linux OS

2.0.1 (2015-05-27)

- Handle new environment variable `PLATFORMIO_BUILD_FLAGS`
- Pass to API requests information about Continuous Integration system. This information will be used by PlatformIO-API.
- Use `include` directories from toolchain when initialising project for IDE ([issue #210](#))
- Added support for new WildFire boards from [Wicked Device](#) to `atmelavr` platform
- Updated [Arduino Framework](#) to 1.6.4 version ([issue #212](#))
- Handle Atmel AVR Symbols when initialising project for IDE ([issue #216](#))
- Fixed bug with converting `*.ino` to `*.cpp`
- Fixed failing with `platformio init --ide eclipse` without boards ([issue #217](#))

2.0.0 (2015-05-22)

Made in Paradise

- PlatformIO as [Continuous Integration \(CI\)](#) tool for embedded projects ([issue #108](#))
- Initialise PlatformIO project for the specified IDE ([issue #151](#))
- PlatformIO CLI 2.0: “platform” related commands have been moved to `platformio platforms` subcommand ([issue #158](#))
- Created PlatformIO [gitter.im](#) room ([issue #174](#))
- Global `-f`, `--force` option which will force to accept any confirmation prompts ([issue #152](#))

- Run project with `platformio run -project-dir` option without changing the current working directory ([issue #192](#))
- Control verbosity of `platformio run` command via `-v`/`--verbose` option
- Add library dependencies for build environment using `lib_install` option in `platformio.ini` ([issue #134](#))
- Specify libraries which are compatible with build environment using `lib_use` option in `platformio.ini` ([issue #148](#))
- Add more boards to PlatformIO project with `platformio init --board` command ([issue #167](#))
- Choose which library to update ([issue #168](#))
- Specify `platformio init --env-prefix` when initialise/update project ([issue #182](#))
- Added new Armstrap boards ([issue #204](#))
- Updated SDK for `espressif` development platform to v1.1 ([issue #179](#))
- Disabled automatic updates by default for platforms, packages and libraries ([issue #171](#))
- Fixed bug with creating copies of source files ([issue #177](#))

1.22.3 PlatformIO 1.0

1.5.0 (2015-05-15)

- Added support of Framework `mbed` for Teensy 3.1 ([issue #183](#))
- Added GDB as alternative uploader to `ststm32` platform ([issue #175](#))
- Added examples with preconfigured IDE projects ([issue #154](#))
- Fixed firmware uploading under Linux OS for Arduino Leonardo board ([issue #178](#))
- Fixed invalid “`mbed`” firmware for Nucleo F411RE ([issue #185](#))
- Fixed parsing of includes for PlatformIO Library Dependency Finder ([issue #189](#))
- Fixed handling symbolic links within source code directory ([issue #190](#))
- Fixed cancelling any previous definition of name, either built in or provided with a `-D` option ([issue #191](#))

1.4.0 (2015-04-11)

- Added `espressif` development platform with ESP01 board
- Integrated PlatformIO with AppVeyor Windows based Continuous Integration system ([issue #149](#))
- Added support for Teensy LC board to `teensy` platform
- Added support for new Arduino based boards by *SparkFun*, *BQ*, *LightUp*, *LowPowerLab*, *Quirkbot*, *RedBear-Lab*, *TinyCircuits* to `atmelavr` platform
- Upgraded `Arduino` Framework to 1.6.3 version ([issue #156](#))
- Upgraded `Energia` Framework to 0101E0015 version ([issue #146](#))
- Upgraded `Arduino` Framework with Teensy Core to 1.22 version ([issue #162](#), [issue #170](#))
- Fixed exceptions with PlatformIO auto-updates when Internet connection isn't active

1.3.0 (2015-03-27)

- Moved PlatformIO source code and repositories from [Ivan Kravets](#) account to PlatformIO Organisation ([issue #138](#))
- Added support for new Arduino based boards by *SparkFun*, *RepRap*, *Sanguino* to [atmelavr](#) platform ([issue #127](#), [issue #131](#))
- Added integration instructions for [Visual Studio](#) and [Sublime Text](#) IDEs
- Improved handling of multi-file *.ino/pde sketches ([issue #130](#))
- Fixed wrong insertion of function prototypes converting *.ino/pde ([issue #137](#), [issue #140](#))

1.2.0 (2015-03-20)

- Added full support of [mbed](#) framework including libraries: *RTOS*, *Ethernet*, *DSP*, *FAT*, *USB*.
- Added [freescalekinetis](#) development platform with Freescale Kinetis Freedom boards
- Added [nordicnrf51](#) development platform with supported boards from *JKSoft*, *Nordic*, *RedBearLab*, *Switch Science*
- Added [nxplpc](#) development platform with supported boards from *CQ Publishing*, *Embedded Artists*, *NGX Technologies*, *NXP*, *Outrageous Circuits*, *SeeedStudio*, *Solder Splash Labs*, *Switch Science*, *u-blox*
- Added support for *ST Nucleo* boards to [ststm32](#) development platform
- Created new [Frameworks](#) page in documentation and added to PlatformIO Web Site ([issue #115](#))
- Introduced online [Embedded Boards Explorer](#)
- Automatically append define `-DPLATFORMIO=%version%` to builder ([issue #105](#))
- Renamed [stm32](#) development platform to [ststm32](#)
- Renamed [opencm3](#) framework to [libopencm3](#)
- Fixed uploading for [atmelsam](#) development platform
- Fixed re-arranging the *.ino/pde files when converting to *.cpp ([issue #100](#))

1.1.0 (2015-03-05)

- Implemented `PLATFORMIO_*` environment variables ([issue #102](#))
- Added support for *SainSmart* boards to [atmelsam](#) development platform
- Added Project Configuration option named `envs_dir`
- Disabled “prompts” automatically for *Continuous Integration* systems ([issue #103](#))
- Fixed firmware uploading for [atmelavr](#) boards which work within `usbtiny` protocol
- Fixed uploading for *Digispark* board ([issue #106](#))

1.0.1 (2015-02-27)

PlatformIO 1.0 - recommended for production

- Changed development status from `beta` to `Production/Stable`
- Added support for *ARM*-based credit-card sized computers: *Raspberry Pi*, *BeagleBone* and *CubieBoard*

- Added `atmelsam` development platform with supported boards: *Arduino Due and Digistump DigiX* ([issue #71](#))
- Added `ststm32` development platform with supported boards: *Discovery kit for STM32L151/152, STM32F303xx, STM32F407/417 lines* and `libOpenCM3 Framework` ([issue #73](#))
- Added `teensy` development platform with supported boards: *Teensy 2.x & 3.x* ([issue #72](#))
- Added new *Arduino* boards to `atmelavr` platform: *Arduino NG, Arduino BT, Arduino Esplora, Arduino Ethernet, Arduino Robot Control, Arduino Robot Motor and Arduino Yun*
- Added support for *Adafruit* boards to `atmelavr` platform: *Adafruit Flora and Adafruit Trinkets* ([issue #65](#))
- Added support for *Digispark* boards to `atmelavr` platform: *Digispark USB Development Board and Digispark Pro* ([issue #47](#))
- Covered code with tests ([issue #2](#))
- Refactored *Library Dependency Finder* ([issues #48, #50, #55](#))
- Added `src_dir` option to `[platformio]` section of `platformio.ini` which allows to redefine location to project's source directory ([issue #83](#))
- Added `--json-output` option to `platformio boards` and `platformio search` commands which allows to return the output in `JSON` format ([issue #42](#))
- Allowed to ignore some libs from *Library Dependency Finder* via `lib_ignore` option
- Improved `platformio run` command: asynchronous output for build process, timing and detailed information about environment configuration ([issue #74](#))
- Output compiled size and static memory usage with `platformio run` command ([issue #59](#))
- Updated *framework-arduino* AVR & SAM to 1.6 stable version
- Fixed an issue with the libraries that are git repositories ([issue #49](#))
- Fixed handling of assembly files ([issue #58](#))
- Fixed compiling error if space is in user's folder ([issue #56](#))
- Fixed *AttributeError: 'module' object has no attribute 'disable_warnings'* when a version of *requests* package is less than 2.4.0
- Fixed bug with invalid process's "return code" when PlatformIO has internal error ([issue #81](#))
- Several bug fixes, increased stability and performance improvements

1.22.4 PlatformIO 0.0

0.10.2 (2015-01-06)

- Fixed an issue with `--json-output` ([issue #42](#))
- Fixed an exception during `platformio upgrade` under Windows OS ([issue #45](#))

0.10.1 (2015-01-02)

- Added `--json-output` option to `platformio list`, `platformio serialports list` and `platformio lib list` commands which allows to return the output in `JSON` format ([issue #42](#))
- Fixed missing auto-uploading by default after `platformio init` command

0.10.0 (2015-01-01)

Happy New Year!

- Implemented `platformio boards` command ([issue #11](#))
- Added support of *Engduino* boards for `atmelavr` platform ([issue #38](#))
- Added `--board` option to `platformio init` command which allows to initialise project with the specified embedded boards ([issue #21](#))
- Added example with uploading firmware via USB programmer (USBasp) for `atmelavr MCUs` ([issue #35](#))
- Automatic detection of port on `platformio serialports monitor` ([issue #37](#))
- Allowed auto-installation of platforms when prompts are disabled ([issue #43](#))
- Fixed `urllib3`'s *SSL* warning under Python <= 2.7.2 ([issue #39](#))
- Fixed bug with *Arduino USB* boards ([issue #40](#))

0.9.2 (2014-12-10)

- Replaced “dark blue” by “cyan” colour for the texts ([issue #33](#))
- Added new setting `enable_prompts` and allowed to disable all *PlatformIO* prompts (useful for cloud compilers) ([issue #34](#))
- Fixed compilation bug on *Windows* with installed *MSVC* ([issue #18](#))

0.9.1 (2014-12-05)

- Ask user to install platform (when it hasn't been installed yet) within `platformio run` and `platformio show` commands
- Improved main `documentation`
- Fixed “*OSError: [Errno 2] No such file or directory*” within `platformio run` command when PlatformIO isn't installed properly
- Fixed example for Eclipse IDE with Tiva board ([issue #32](#))
- Upgraded Eclipse Project Examples to latest *Luna* and *PlatformIO* releases

0.9.0 (2014-12-01)

- Implemented `platformio settings` command
- Improved `platformio init` command. Added new option `--project-dir` where you can specify another path to directory where new project will be initialized ([issue #31](#))
- Added *Migration Manager* which simplifies process with upgrading to a major release
- Added *Telemetry Service* which should help us make *PlatformIO* better
- Implemented *PlatformIO AppState Manager* which allow to have multiple `.platformio` states.
- Refactored *Package Manager*
- Download Manager: fixed SHA1 verification within *Cygwin Environment* ([issue #26](#))
- Fixed bug with code builder and built-in Arduino libraries ([issue #28](#))

0.8.0 (2014-10-19)

- Avoided trademark issues in `library.json` with the new fields: `frameworks`, `platforms` and `dependencies` (issue #17)
- Switched logic from “Library Name” to “Library Registry ID” for all `platformio lib` commands (`install`, `uninstall`, `update` and etc.)
- Renamed `author` field to `authors` and allowed to setup multiple authors per library in `library.json`
- Added option to specify “maintainer” status in `authors` field
- New filters/options for `platformio lib search` command: `--framework` and `--platform`

0.7.1 (2014-10-06)

- Fixed bug with order for includes in conversation from INO/PDE to CPP
- Automatic detection of port on upload (issue #15)
- Fixed lib update crashing when no libs are installed (issue #19)

0.7.0 (2014-09-24)

- Implemented new `[platformio]` section for Configuration File with `home_dir` option (issue #14)
- Implemented *Library Manager* (issue #6)

0.6.0 (2014-08-09)

- Implemented `platformio serialports monitor` (issue #10)
- Fixed an issue `ImportError: No module named platformio.util` (issue #9)
- Fixed bug with auto-conversation from Arduino *.ino to *.cpp

0.5.0 (2014-08-04)

- Improved nested lookups for libraries
- Disabled default warning flag “`-Wall`”
- Added auto-conversation from *.ino to valid *.cpp for Arduino/Energia frameworks (issue #7)
- Added Arduino example with external library (*Adafruit CC3000*)
- Implemented `platformio upgrade` command and “auto-check” for the latest version (issue #8)
- Fixed an issue with “auto-reset” for *Raspduino* board
- Fixed a bug with nested libs building

0.4.0 (2014-07-31)

- Implemented `platformio serialports` command
- Allowed to put special build flags only for `src` files via `src_build_flags` environment option
- Allowed to override some of settings via system environment variables such as: `PLATFORMIO_SRC_BUILD_FLAGS` and `PLATFORMIO_ENVS_DIR`
- Added `--upload-port` option for `platformio run` command
- Implemented (especially for SmartAnthill) `platformio run -t uploadlazy` target (no dependencies to framework libs, ELF and etc.)
- Allowed to skip default packages via `platformio install --skip-default-package` option
- Added tools for *Raspberry Pi* platform
- Added support for *Microduino* and *Raspduino* boards in `atmelavr` platform

0.3.1 (2014-06-21)

- Fixed auto-installer for Windows OS (bug with %PATH% custom installation)

0.3.0 (2014-06-21)

- Allowed to pass multiple “SomePlatform” to install/uninstall commands
- Added “IDE Integration” section to README with Eclipse project examples
- Created auto installer script for *PlatformIO* ([issue #3](#))
- Added “Super-Quick” way to Installation section (README)
- Implemented “build_flags” option for environments ([issue #4](#))

0.2.0 (2014-06-15)

- Resolved issue #1 “Build referred libraries”
- Renamed project’s “libs” directory to “lib”
- Added `arduino-internal-library` example
- Changed to beta status

0.1.0 (2014-06-13)

- Birth! First alpha release

1.23 Migrating from 2.x to 3.0

Guidance on how to upgrade from PlatformIO v2.x to v3.x with emphasis on major changes, what is new, and what is been removed.

PlatformIO 3 is not backwards compatible with v2.x. Use this section as a general guide to upgrading from v2.x to v3.0. For a broader overview, see [what is new](#) in the v3.0 release announcement.

Contents

- *Major PlatformIO CLI changes*
- *What is new*
 - *Development Platforms*
 - *Library Manager and Intelligent Build System*
 - *Command Line Interface*
 - Project Configuration File `platformio.ini`
- *What is removed*
 - *Command Line Interface*
 - Project Configuration File `platformio.ini`

1.23.1 Major PlatformIO CLI changes

Note: PlatformIO 3.x is 100% non-blocking! You do not need to use `--force` option or setup special `PLATOFMRIO_SETTING_ENABLE_PROMPTS` environment. Use PlatformIO 3.0 with sub-processing without any risk!

This table shows the CLI changes between v2.x and v3.0.

PlatformIO 2.x	PlatformIO 3.x
<code>platformio platforms</code>	<i>platformio platform</i>
<code>platformio serialports</code>	<i>platformio device</i>
<code>platformio settings set enable_prompts false</code>	Removed! Now, all PlatformIO 3.0 CLI is 100% non-blocking!

PlatformIO 2.x commands will be converted to PlatformIO 3.x automatically. Nevertheless, we recommend to use PlatformIO 3.x commands for the new projects.

1.23.2 What is new

Development Platforms

We have introduced *Manifest File `platform.json`* and ported PlatformIO 2.x development platforms according PlatformIO 3.0 decentralized architecture. Now, platform related things (build scripts, LD scripts, board configs, package requirements) are located in own repository. Here is the full list with [PlatformIO 3.0 open source development platforms](#). You can fork it, modify or create custom. See [Custom Development Platform](#) guide for details.

- *Manifest File `platform.json`*
- `espressif` development platform has been renamed to [*Espressif 8266*](#)
- PlatformIO 3.0 *Platform Manager*
- Custom package repositories
- External embedded board configuration files, isolated build scripts

- Embedded Board compatibility with more than one development platform

Library Manager and Intelligent Build System

- Powerful and super-fast *Library Dependency Finder (LDF)* that interprets C/C++ Preprocessor conditional macros with deep search behavior
- Project dependencies per build environment using *projectconf.lib_deps* option
- Depend on a library using VCS URL (GitHub, Git, ARM mbed code registry, Hg, SVN)
- Install library by name
- Strict search for library dependencies
- Multiple library storages: Project's Local, PlatformIO's Global or Custom
- Allowed *library.json* to specify sources other than PlatformIO's Repository
- Check library compatibility with project environment before building
- Control Library Dependency Finder for compatibility using *lib_compat_mode* option
- Custom library storages/directories with *lib_extra_dirs* option
- Handle extra build flags, source filters and build script from *library.json*
- Allowed to disable library archiving (*.ar)
- Show detailed build information about dependent libraries (Library Dependency Graph)
- Support for the 3rd party manifests (Arduino IDE "library.properties" and ARM mbed "module.json")
- Build System: Attach custom Before/Pre and After/Post actions for targets using *extra_scripts*

Command Line Interface

We have added new commands and changed some existing ones. Here are the new or updated commands and options.

Command	Description
<code>platformio boards</code>	Returns all supported boards by PlatformIO
<code>platformio boards --installed</code>	Returns currently installed boards
<code>platformio ci --project-option</code>	Pass custom option from <i>Project Configuration File platformio.ini</i>
<code>platformio ci --verbose</code>	Print detailed information about build process
<code>platformio init --project-option</code>	Pass custom option from <i>Project Configuration File platformio.ini</i>
<code>platformio lib --global</code>	Manage PlatformIO <i>Global Library Storage</i>
<code>platformio lib --storage-dir</code>	Manage <i>Custom Library Storage</i>
<code>platformio lib install</code>	New PlatformIO 3.0 Library Manager! Semantic Versioning, VCS support and external URL support
<code>platformio lib install --silent</code>	Suppress progress reporting when install library
<code>platformio lib install --interactive</code>	Allow to make a choice for all prompts when install library
<code>platformio lib search --header</code>	Search library by specific header file name (include)
<code>platformio lib update --only-check</code>	Do not update, only check for new version
<code>platformio platform</code>	New PlatformIO 3.0 Platform Manager! Semantic Versioning, VCS support and external URL support.
<code>platformio platform update --only-packages</code>	Update only platform packages
<code>platformio platform update --only-check</code>	Do not update, only check for new version
<code>platformio run</code>	By default, prints only human-readable information when processing environments
<code>platformio run --verbose</code>	Print detailed processing information
<code>platformio settings set force_verbose true</code>	Force verbose output when processing environments (globally)
<code>platformio test</code>	PIO Plus Unit Testing
<code>platformio update --only-check</code>	Do not update, only check for new version

Project Configuration File `platformio.ini`

We have added new options and changed some existing ones. Here are the new or updated options.

Section	Option	Description
platformio	<i>libdeps_dir</i>	Internal storage where <i>Library Manager</i> will install project dependencies
platformio	<i>test_dir</i>	Directory where <i>PIO Unit Testing</i> engine will look for the tests
env	<i>lib_deps</i>	Specify project dependencies that should be installed automatically to <i>libdeps_dir</i> before environment processing.
env	<i>platform</i>	PlatformIO 3.0 allows to use specific version of platform using Semantic Versioning (X.Y.Z=MAJOR.MINOR.PATCH).
env	<i>lib_extra_dirs</i>	A list with extra directories/storages where <i>Library Dependency Finder (LDF)</i> will look for dependencies
env	<i>lib_ldf_mode</i>	This option specifies how does <i>Library Dependency Finder (LDF)</i> should analyze dependencies (#include directives)
env	<i>lib_compat_mode</i>	Library compatibility mode allows to control strictness of <i>Library Dependency Finder (LDF)</i>
env	<i>test_ignore</i>	Ignore tests where the name matches specified patterns

1.23.3 What is removed

Command Line Interface

The following commands have been dropped or changed in v3.0.

Command	Description
platformio init --enable-auto-uploading	Use <code>platformio init --project-option</code> instead of it with targets = upload value

Project Configuration File `platformio.ini`

The following options have been dropped or changed in v3.0.

Section	Op-tion	Description
platformio	<i>lib_dir</i>	Changed: Project's own/private libraries, where in PlatformIO 2.x it was global library storage

Bibliography

[Embedds] Embedds.com: Develop easier with PlatformIO ecosystem

Symbols

- quiet
 - platformio-device-monitor command line option, 35
 - platformio-remote-device-monitor command line option, 91
- build-dir
 - platformio-ci command line option, 29
- core-packages
 - platformio-update command line option, 116
- dev
 - platformio-upgrade command line option, 120
- disable-auto-clean
 - platformio-remote-run command line option, 94
 - platformio-run command line option, 103
- dtr
 - platformio-device-monitor command line option, 35
 - platformio-remote-device-monitor command line option, 91
- echo
 - platformio-device-monitor command line option, 35
 - platformio-remote-device-monitor command line option, 91
- encoding
 - platformio-device-monitor command line option, 35
 - platformio-remote-device-monitor command line option, 91
- env-prefix
 - platformio-init command line option, 39
- eol
 - platformio-device-monitor command line option, 35
 - platformio-remote-device-monitor command line option, 91
- exclude
 - platformio-ci command line option, 29
- exit-char
 - platformio-device-monitor command line option, 35
 - platformio-remote-device-monitor command line option, 91
- force, -f
 - platformio-command-line option, 20
- help, -h
 - platformio-command-line option, 20
- host
 - platformio-home command line option, 37
- id
 - platformio-lib-search command line option, 55
- ide
 - platformio-init command line option, 39
- installed
 - platformio-boards command line option, 26
- interactive
 - platformio-lib-install command line option, 46
- interface
 - platformio-debug command line option, 30
- json-output
 - platformio-account-show command line option, 24
 - platformio-account-token command line option, 25
 - platformio-boards command line option, 26
 - platformio-device-list command line option, 32
 - platformio-lib-builtin command line option, 41
 - platformio-lib-list command line option, 50
 - platformio-lib-search command line option, 56
 - platformio-lib-show command line option, 62, 64
 - platformio-lib-update command line option, 70
 - platformio-platform-frameworks command line option, 71
 - platformio-platform-list command line option, 78
 - platformio-platform-search command line option, 79
 - platformio-platform-update command line option, 85
 - platformio-remote-device-list command line option, 89
- keep-build-dir
 - platformio-ci command line option, 29
- logical
 - platformio-device-list command line option, 32
- mdns
 - platformio-device-list command line option, 32
- menu-char
 - platformio-command-line option, 20

platformio-device-monitor command line option, [35](#)
platformio-remote-device-monitor command line option, [91](#)
-monitor-dtr
 platformio-test command line option, [115](#)
-monitor-rts
 platformio-test command line option, [115](#)
-no-open
 platformio-home command line option, [38](#)
-no-reset
 platformio-test command line option, [115](#)
-page
 platformio-lib-search command line option, [56](#)
-parity
 platformio-device-monitor command line option, [35](#)
 platformio-remote-device-monitor command line option, [90](#)
-password, -p
 platformio-account-login command line option, [22](#)
-port
 platformio-home command line option, [37](#)
-raw
 platformio-device-monitor command line option, [35](#)
 platformio-remote-device-monitor command line option, [91](#)
-regenerate
 platformio-account-token command line option, [25](#)
-rts
 platformio-device-monitor command line option, [35](#)
 platformio-remote-device-monitor command line option, [91](#)
-rtscs
 platformio-device-monitor command line option, [35](#)
 platformio-remote-device-monitor command line option, [90](#)
-serial
 platformio-device-list command line option, [32](#)
-skip-default
 platformio-platform-install command line option, [74](#)
-storage
 platformio-lib-builtin command line option, [41](#)
-test-port
 platformio-remote-test command line option, [97](#)
 platformio-test command line option, [115](#)
-upload-port
 platformio-remote-run command line option, [94](#)
 platformio-remote-test command line option, [97](#)
 platformio-run command line option, [103](#)
 platformio-test command line option, [115](#)
-username, -u
 platformio-account-forgot command line option, [22](#)
 platformio-account-login command line option, [22](#)
 platformio-account-register command line option, [24](#)
-version
 platformio command line option, [20](#)
-with-package
 platformio-platform-install command line option, [74](#)
-without-building
 platformio-remote-test command line option, [97](#)
 platformio-test command line option, [115](#)
-without-package
 platformio-platform-install command line option, [74](#)
-without-uploading
 platformio-remote-test command line option, [97](#)
 platformio-test command line option, [115](#)
-xonxoff
 platformio-device-monitor command line option, [35](#)
 platformio-remote-device-monitor command line option, [91](#)
-O, -project-option
 platformio-ci command line option, [29](#)
 platformio-init command line option, [39](#)
-P, -project-conf
 platformio-ci command line option, [29](#)
-a, -author
 platformio-lib-search command line option, [55](#)
-b, -baud
 platformio-device-monitor command line option, [34](#)
 platformio-remote-device-monitor command line option, [90](#)
-b, -board
 platformio-ci command line option, [29](#)
 platformio-init command line option, [39](#)
-c, -only-check
 platformio-lib-update command line option, [70](#)
 platformio-platform-update command line option, [85](#)
 platformio-remote-update command line option, [98](#)
 platformio-update command line option, [116](#)
-d, -project-dir
 platformio-debug command line option, [30](#)
 platformio-device-monitor command line option, [36](#)
 platformio-init command line option, [39](#)
 platformio-remote-run command line option, [94](#)
 platformio-remote-test command line option, [97](#)
 platformio-run command line option, [103](#)
 platformio-test command line option, [115](#)
-d, -storage-dir
 platformio-lib command line option, [40](#)
-d, -working-dir
 platformio-remote-agent-start command line option, [87](#)
-e, -environment
 platformio-debug command line option, [30](#)
 platformio-device-monitor command line option, [36](#)
 platformio-remote-run command line option, [93](#)
 platformio-remote-test command line option, [96](#)

platformio-run command line option, 102
 platformio-test command line option, 114
-f, --filter
 platformio-device-monitor command line option, 35
 platformio-remote-device-monitor command line option, 91
 platformio-test command line option, 114
-f, --force
 platformio-lib-install command line option, 46
 platformio-platform-install command line option, 74
-f, --framework
 platformio-lib-search command line option, 55
-g, --global
 platformio-lib command line option, 40
-i, --header
 platformio-lib-search command line option, 56
-i, --ignore
 platformio-remote-test command line option, 96
 platformio-test command line option, 114
-k, --keyword
 platformio-lib-search command line option, 55
-l, --lib
 platformio-ci command line option, 29
-n, --name
 platformio-lib-search command line option, 55
 platformio-remote-agent-start command line option, 87
-p, --only-packages
 platformio-platform-update command line option, 85
-p, --platform
 platformio-lib-search command line option, 55
-p, --port
 platformio-device-monitor command line option, 34
 platformio-remote-device-monitor command line option, 90
-r, --force-remote
 platformio-remote-run command line option, 94
 platformio-remote-test command line option, 97
-s, --share
 platformio-remote-agent-start command line option, 87
-s, --silent
 platformio-init command line option, 39
 platformio-lib-install command line option, 46
 platformio-run command line option, 103
-t, --target
 platformio-remote-run command line option, 93
 platformio-run command line option, 102
-v, --verbose
 platformio-ci command line option, 29
 platformio-debug command line option, 31
 platformio-remote-run command line option, 94
 platformio-remote-test command line option, 97

platformio-run command line option, 103
 platformio-test command line option, 115

E

environment variable

CI, 190
 PLATFORMIO_AUTH_TOKEN, 22, 25, 191, 490, 648
 PLATFORMIO_BOARDS_DIR, 169, 191
 PLATFORMIO_BUILD_DIR, 167, 191
 PLATFORMIO_BUILD_FLAGS, 172, 191, 639, 649
 PLATFORMIO_DATA_DIR, 168, 191
 PLATFORMIO_DISABLE_PROGRESSBAR, 191
 PLATFORMIO_EXTRA_SCRIPTS, 192, 194
 PLATFORMIO_FORCE_COLOR, 191
 PLATFORMIO_HOME_DIR, 166, 191, 664
 PLATFORMIO_INCLUDE_DIR, 167, 191
 PLATFORMIO_LIB_DIR, 167, 191
 PLATFORMIO_LIB_EXTRA_DIRS, 180, 192
 PLATFORMIO_LIBDEPS_DIR, 168, 191
 PLATFORMIO_REMOTE_AGENT_DIR, 191
 PLATFORMIO_SETTING_AUTO_UPDATE_LIBRARIES, 192
 PLATFORMIO_SETTING_AUTO_UPDATE_PLATFORMS, 192
 PLATFORMIO_SETTING_CHECK_LIBRARIES_INTERVAL, 192
 PLATFORMIO_SETTING_CHECK_PLATFORMIO_INTERVAL, 192
 PLATFORMIO_SETTING_CHECK_PLATFORMS_INTERVAL, 192
 PLATFORMIO_SETTING_ENABLE_CACHE, 192
 PLATFORMIO_SETTING_ENABLE_SSL, 193
 PLATFORMIO_SETTING_ENABLE_TELEMETRY, 193
 PLATFORMIO_SETTING_FORCE_VERBOSE, 30, 31, 94, 97, 103, 115, 193
 PLATFORMIO_SETTING_PROJECTS_DIR, 193
 PLATFORMIO_SRC_BUILD_FLAGS, 174, 192
 PLATFORMIO_SRC_DIR, 167, 191
 PLATFORMIO_SRC_FILTER, 175, 192
 PLATFORMIO_TEST_DIR, 168, 191
 PLATFORMIO_UPLOAD_FLAGS, 176, 192
 PLATFORMIO_UPLOAD_PORT, 175, 192

P

platformio command line option

-force, -f, 20
 -help, -h, 20
 -version, 20

platformio-account-forgot command line option
 -username, -u, 22

platformio-account-login command line option
 –password, -p, 22
 –username, -u, 22

platformio-account-register command line option
 –username, -u, 24

platformio-account-show command line option
 –json-output, 24

platformio-account-token command line option
 –json-output, 25
 –regenerate, 25

platformio-boards command line option
 –installed, 26
 –json-output, 26

platformio-ci command line option
 –build-dir, 29
 –exclude, 29
 –keep-build-dir, 29
 –O, –project-option, 29
 –P, –project-conf, 29
 –b, –board, 29
 –l, –lib, 29
 –v, –verbose, 29

platformio-debug command line option
 –interface, 30
 –d, –project-dir, 30
 –e, –environment, 30
 –v, –verbose, 31

platformio-device-list command line option
 –json-output, 32
 –logical, 32
 –mdns, 32
 –serial, 32

platformio-device-monitor command line option
 –quiet, 35
 –dtr, 35
 –echo, 35
 –encoding, 35
 –eol, 35
 –exit-char, 35
 –menu-char, 35
 –parity, 35
 –raw, 35
 –rts, 35
 –rtscts, 35
 –xonxoff, 35
 –b, –baud, 34
 –d, –project-dir, 36
 –e, –environment, 36
 –f, –filter, 35
 –p, –port, 34

platformio-home command line option
 –host, 37
 –no-open, 38
 –port, 37

platformio-init command line option
 –env-prefix, 39
 –ide, 39
 –O, –project-option, 39
 –b, –board, 39
 –d, –project-dir, 39
 –s, –silent, 39

platformio-lib command line option
 –d, –storage-dir, 40
 –g, –global, 40

platformio-lib-builtin command line option
 –json-output, 41
 –storage, 41

platformio-lib-install command line option
 –interactive, 46
 –f, –force, 46
 –s, –silent, 46

platformio-lib-list command line option
 –json-output, 50

platformio-lib-search command line option
 –id, 55
 –json-output, 56
 –page, 56
 –a, –author, 55
 –f, –framework, 55
 –i, –header, 56
 –k, –keyword, 55
 –n, –name, 55
 –p, –platform, 55

platformio-lib-show command line option
 –json-output, 62, 64

platformio-lib-update command line option
 –json-output, 70
 –c, –only-check, 70

platformio-platform-frameworks command line option
 –json-output, 71

platformio-platform-install command line option
 –skip-default, 74
 –with-package, 74
 –without-package, 74
 –f, –force, 74

platformio-platform-list command line option
 –json-output, 78

platformio-platform-search command line option
 –json-output, 79

platformio-platform-update command line option
 –json-output, 85
 –c, –only-check, 85
 –p, –only-packages, 85

platformio-remote-agent-start command line option
 –d, –working-dir, 87
 –n, –name, 87
 –s, –share, 87

platformio-remote-device-list command line option

- json-output, 89
- platformio-remote-device-monitor command line option
 - quiet, 91
 - dtr, 91
 - echo, 91
 - encoding, 91
 - eol, 91
 - exit-char, 91
 - menu-char, 91
 - parity, 90
 - raw, 91
 - rts, 91
 - rtscts, 90
 - xonxoff, 91
 - b, —baud, 90
 - f, —filter, 91
 - p, —port, 90
- platformio-remote-run command line option
 - disable-auto-clean, 94
 - upload-port, 94
 - d, —project-dir, 94
 - e, —environment, 93
 - r, —force-remote, 94
 - t, —target, 93
 - v, —verbose, 94
- platformio-remote-test command line option
 - test-port, 97
 - upload-port, 97
 - without-building, 97
 - without-uploading, 97
 - d, —project-dir, 97
 - e, —environment, 96
 - i, —ignore, 96
 - r, —force-remote, 97
 - v, —verbose, 97
- platformio-remote-update command line option
 - c, —only-check, 98
- platformio-run command line option
 - disable-auto-clean, 103
 - upload-port, 103
 - d, —project-dir, 103
 - e, —environment, 102
 - s, —silent, 103
 - t, —target, 102
 - v, —verbose, 103
- platformio-test command line option
 - monitor-dtr, 115
 - monitor-rts, 115
 - no-reset, 115
 - test-port, 115
 - upload-port, 115
 - without-building, 115
 - without-uploading, 115
 - d, —project-dir, 115
- e, —environment, 114
- f, —filter, 114
- i, —ignore, 114
- v, —verbose, 115
- platformio-update command line option
 - core-packages, 116
 - c, —only-check, 116
- platformio-upgrade command line option
 - dev, 120
- PLATFORMIO_AUTH_TOKEN, 22, 25, 490, 648
- PLATFORMIO_BOARDS_DIR, 169
- PLATFORMIO_BUILD_DIR, 167
- PLATFORMIO_BUILD_FLAGS, 172, 639, 649
- PLATFORMIO_DATA_DIR, 168
- PLATFORMIO_DISABLE_PROGRESSBAR, 191
- PLATFORMIO_EXTRA_SCRIPTS, 194
- PLATFORMIO_HOME_DIR, 166, 664
- PLATFORMIO_INCLUDE_DIR, 167
- PLATFORMIO_LIB_DIR, 167
- PLATFORMIO_LIB_EXTRA_DIRS, 180
- PLATFORMIO_LIBDEPS_DIR, 168
- PLATFORMIO_SETTING_FORCE_VERBOSE, 30, 31, 94, 97, 103, 115
- PLATFORMIO_SRC_BUILD_FLAGS, 174
- PLATFORMIO_SRC_DIR, 167
- PLATFORMIO_SRC_FILTER, 175
- PLATFORMIO_TEST_DIR, 168
- PLATFORMIO_UPLOAD_FLAGS, 176
- PLATFORMIO_UPLOAD_PORT, 175