platformIO

Visual studio code plugin

Leo Kuipers – l.a.Kuipers@saxion.nl

What is platformIO?

- In short:
  - Embedded C/C++ development toolset built on top of Microsoft's Visual Studio Code
- In marketing:
  - PlatformIO is a cross-platform, cross-architecture, multiple framework, professional tool for embedded systems engineers and for software developers who write applications for embedded products.

In other words:

- Cross-platform IDE which supports many different C/C++ software development kits (SDKs, called Frameworks in platformIO) and includes a lot of sophisticated developer tools.

Why platformIO?

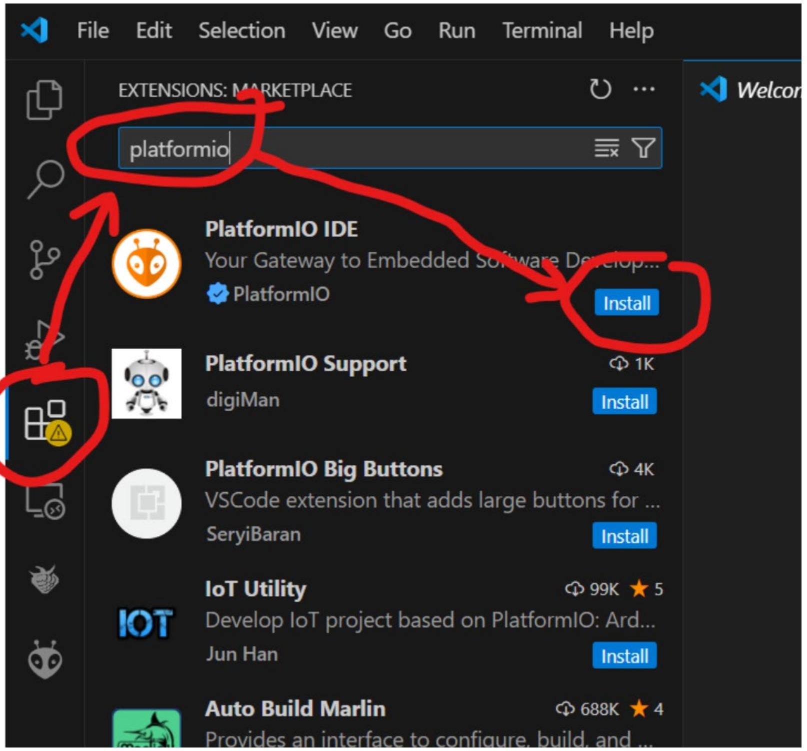
- It just works better.
- Arduino IDE is a meh editor.
- Visual Studio Code is a proper IDE.
- VS Code “Extensions” add a lot of functionality. E.g.:
  - C/C++ Intellisense → = Autocomplete!
  - Git directly from within the editor
  - Github Copilot → Use Copilot AI to help you to program code
- PlatformIO adds embedded programming to Visual Studio Code.

SDK? Framework?

- You don’t want to start from scratch. Really…
- You want to use a toolbox filled with tools that are known to work
    - Libraries with pre-written code so you don’t have to write it again
    - Compilers/linkers for the specific microcontroller that you use
    - Connectivity tools to upload your code to the microcontroller
    - Etc.
- In software terms, that toolbox is an SDK: Software Development Kit.
- There are many SDKs for microcontrollers. Some are specific to a microcontroller family. Some are general purpose, like Arduino SDK.
- PlatformIO calls this a Framework.

How to install platformIO?

- Install visual studio code: code.visualstudio.com
  - Make sure also to install the C/C++ Intellisense extension
- Install PlatformIO plugin for visual studio code



platformio

PlatformIO IDE
Your Gateway to Embedded Software Develop...
PlatformIO
Install

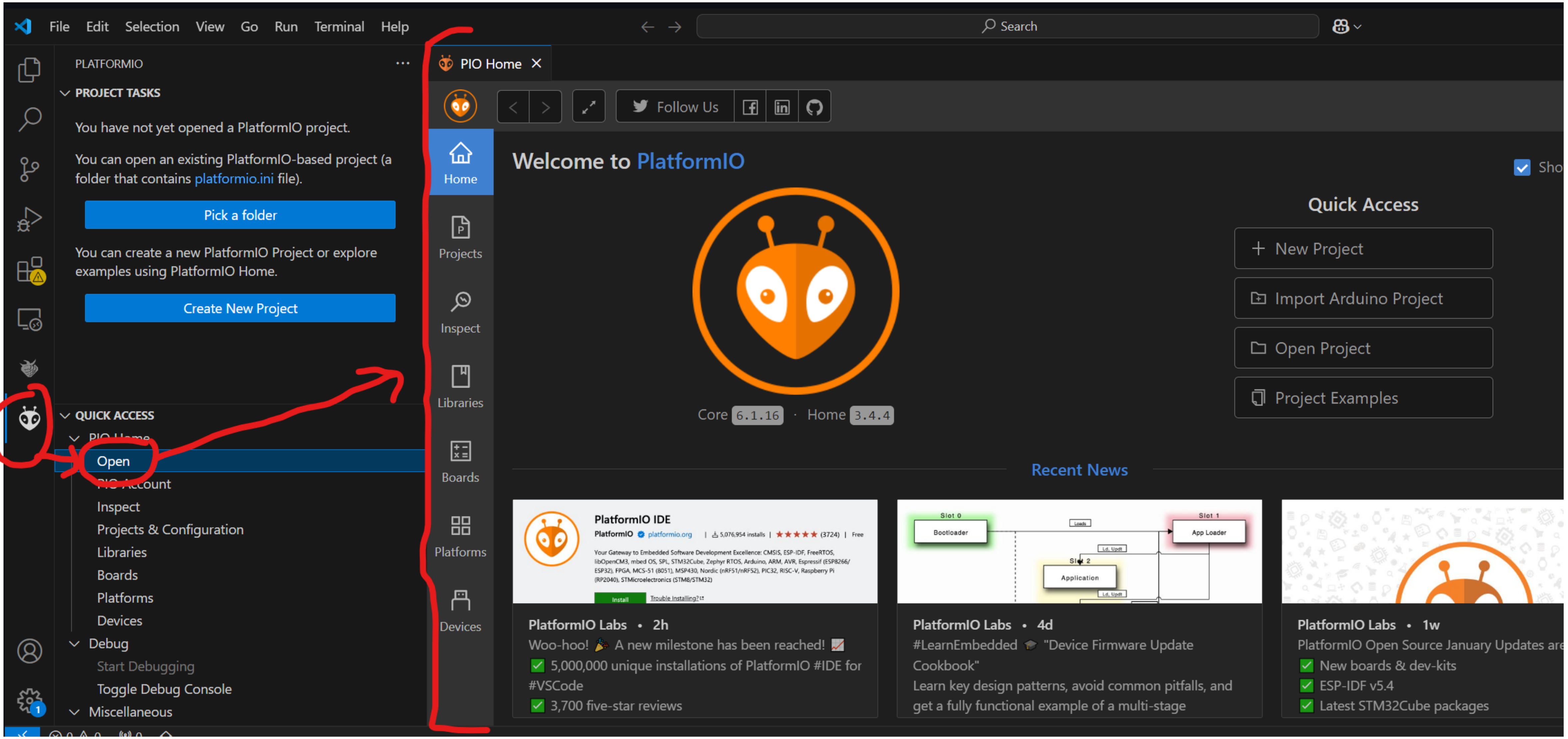
PlatformIO Support 1K
digiMan
Install

PlatformIO Big Buttons 4K
VSCode extension that adds large buttons for ...
SeryiBaran
Install

IoT Utility 99K ★ 5
Develop IoT project based on PlatformIO: Ard...
Jun Han
Install

Auto Build Marlin 688K ★ 4
Provides an interface to configure, build, and ...

How to “open” platformIO?



File Edit Selection View Go Run Terminal Help

Search

PLATFORMIO

PROJECT TASKS

You have not yet opened a PlatformIO project.

You can open an existing PlatformIO-based project (a folder that contains platformio.ini file).

Pick a folder

You can create a new PlatformIO Project or explore examples using PlatformIO Home.

Create New Project

QUICK ACCESS

- PIO Home
  - Open
  - PIO Account
  - Inspect
  - Projects & Configuration
  - Libraries
  - Boards
  - Platforms
  - Devices
- Debug
  - Start Debugging
  - Toggle Debug Console
- Miscellaneous

PIO Home

Follow Us

Home · Projects · Inspect · Libraries · Boards · Platforms · Devices

Welcome to PlatformIO

Core 6.1.16 · Home 3.4.4

Quick Access

- New Project
- Import Arduino Project
- Open Project
- Project Examples

Recent News

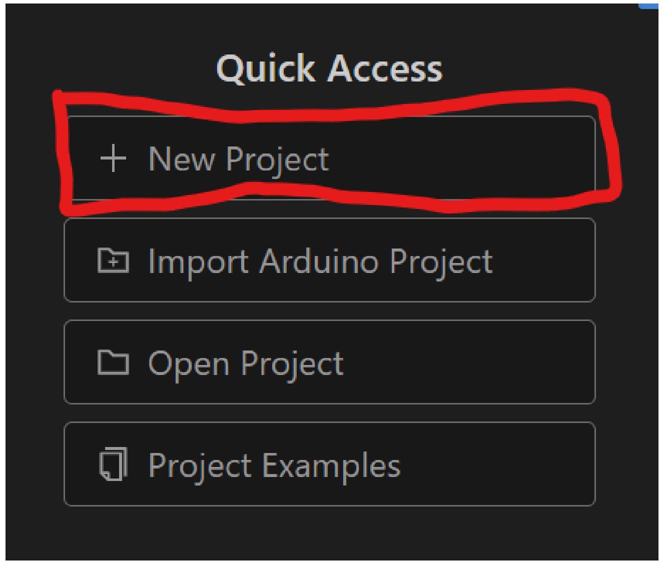
PlatformIO Labs • 2h
Woo-hoo! 🎉 A new milestone has been reached! 📈
✅ 5,000,000 unique installations of PlatformIO #IDE for #VSCode
✅ 3,700 five-star reviews

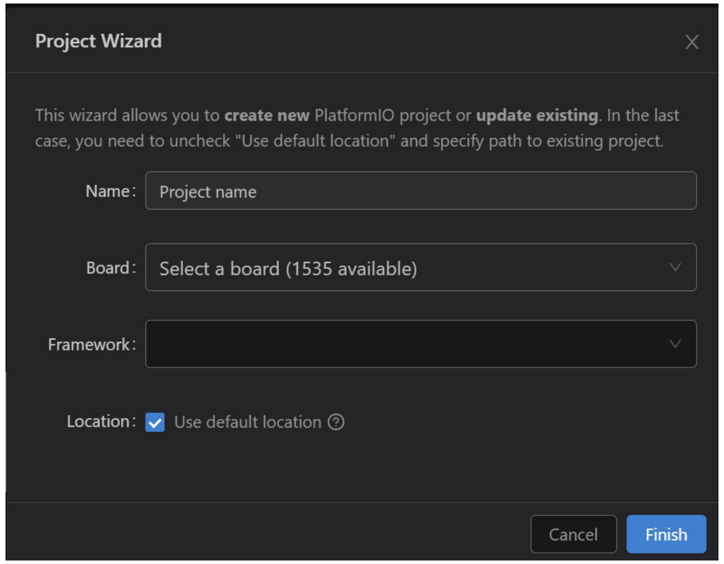
PlatformIO Labs • 4d
#LearnEmbedded 🎓 "Device Firmware Update Cookbook"
Learn key design patterns, avoid common pitfalls, and get a fully functional example of a multi-stage

PlatformIO Labs • 1w
PlatformIO Open Source January Updates are
✅ New boards & dev-kits
✅ ESP-IDF v5.4
✅ Latest STM32Cube packages

Create a project

- Create a new project
- Give it a name
- Select your microcontroller
- Select the framework you want to use




Project Wizard

This wizard allows you to **create new** PlatformIO project or **update existing**. In the last case, you need to uncheck "Use default location" and specify path to existing project.

Name: Project name

Board: Select a board (1535 available)

Framework:

Location: ✓ Use default location

platformio.ini

- Creating a platformIO project will generate a platformio.ini file
- Core settings are defined in this file (e.g. which board, which framework)
- You can always manually change these settings (e.g. when you want the same code to run on a different board)

RP2040 with Arduino framework

Using platformIO

Raspberry Pi Pico RP2040 and platformIO

- Can I use RP2040 with Arduino framework on platformIO?
  - Yes, but…
- For RP2040 there are 2 Arduino frameworks
  1. Official ArduinoCore-mbed from Arduino based on Arduino APIs, running on top of Mbed OS
     - Mbed platform will reach end-of-life in july 2026
     - Source https://github.com/platformio/platform-raspberrypi/issues/66
  2. Community-made Earle Philhower's Arduino-Pico based on Arduino APIs, using official Raspberry Pi Pico SDK
     - Not officially supported by platformIO
     - Source https://community.platformio.org/t/request-to-add-platformio-support-for-earle-philhowers-arduino-pico-raspberry-pi-pico-sdk/22285
- So you want to use Earle Philhower's Arduino Pico!

Raspberry Pi Pico RP2040 and platformIO

- Will there be official support for Earle Philhower’s Arduino Pico?
    - platformIO is a for-profit organization
    - Their business model is that silicon manufacturers pay them to get their microcontrollers officially supported in platformIO.
    - Raspberry Pi ltd. doesn’t want to pay the fee, so no official support.

  Source https://github.com/platformio/platform-raspberrypi/pull/36

- Fortunately the community knows how to deal with this:

https://arduino-pico.readthedocs.io/en/latest/platformio.html

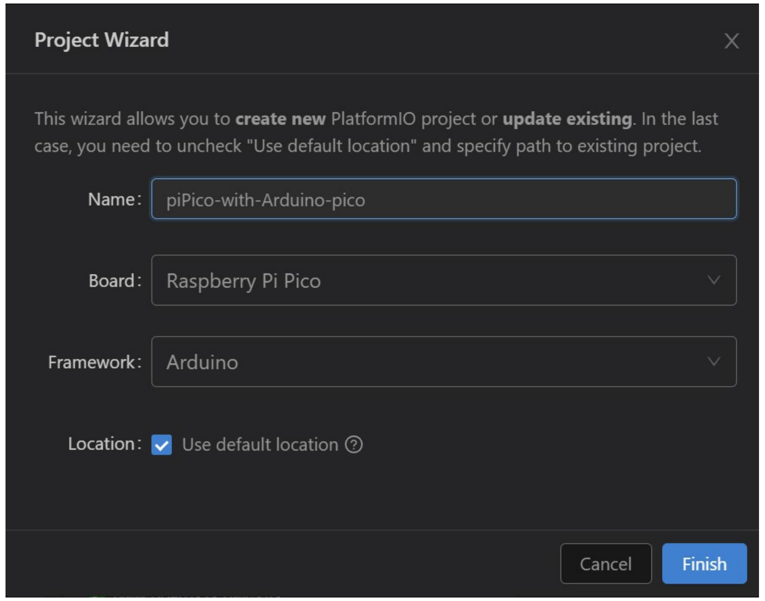
But first: fix Windows

- By default, Windows has a limited path length that is not long enough
- Step 0: Install Git for Windows https://git-scm.com/downloads/win
- Step 1: Enable long paths in git
- Step 2: Enable long paths in Windows OS
- Step 3: Reboot

- See: https://arduino-pico.readthedocs.io/en/latest/platformio.html#important-steps-for-windows-users-before-installing

Raspberry Pi Pico RP2040 and platformIO

- Create a new project
- Select Raspberry Pi Pico
- Select Arduino
- Click Finish
- Change the generated platformio.ini file


Project Wizard

This wizard allows you to **create new** PlatformIO project or **update existing**. In the last case, you need to uncheck "Use default location" and specify path to existing project.

Name: piPico-with-Arduino-pico

Board: Raspberry Pi Pico

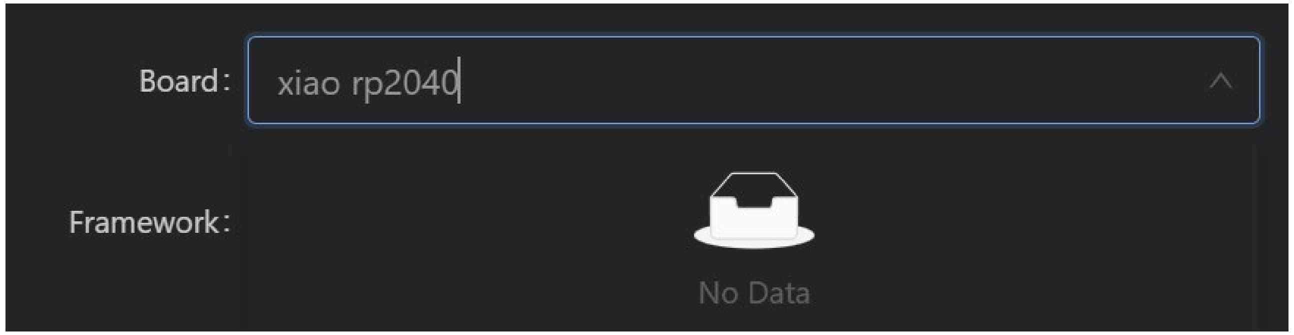
Framework: Arduino

Location: ✓ Use default location

```
[env:pico]
platform = https://github.com/maxgerhardt/platform-raspberrypi.git
board = pico
framework = arduino
board_build.core = earlephilhower
```

Xiao RP2040 + Arduino with platformIO

- Where is Xiao RP2040?



Board: xiao rp2040

Framework:

No Data

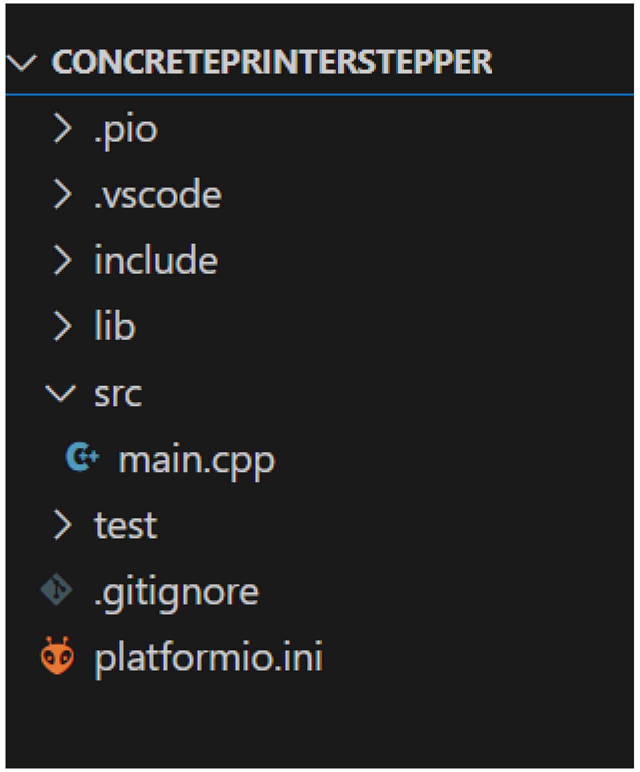
- Solution: Create a project for raspberry pi pico and change platformio.ini

```
[env]
platform = https://github.com/maxgerhardt/platform-raspberrypi.git
framework = arduino
board_build.core = earlephilhower
board_build.filesystem_size = 0.5m
[env:seeed_xiao_rp2040]
board = seeed_xiao_rp2040
```

Project folder overview

- src folder: place your source code here
- lib folder: place for project specific libraries

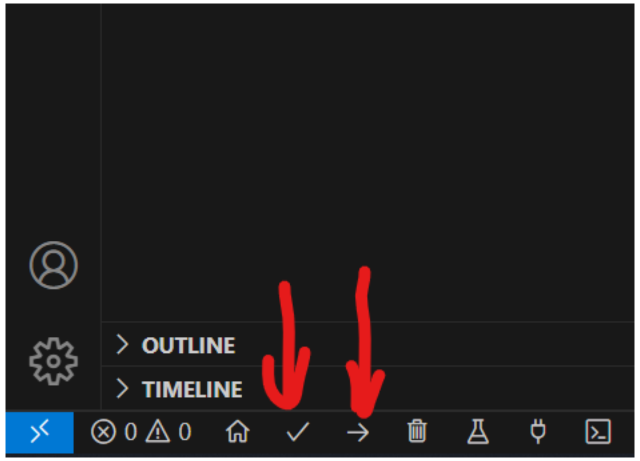
- main.cpp will be your main program!



```
CONCRETEPRINTERSTEPPER
  > .pio
  > .vscode
  > include
  > lib
  v src
     main.cpp
  > test
    .gitignore
    platformio.ini
```

Compile & run

- Build & upload commands can be found in the bottom-left bar:



- Upload will try to automatically discover the upload port. You can manually define the upload port in platformio.ini:

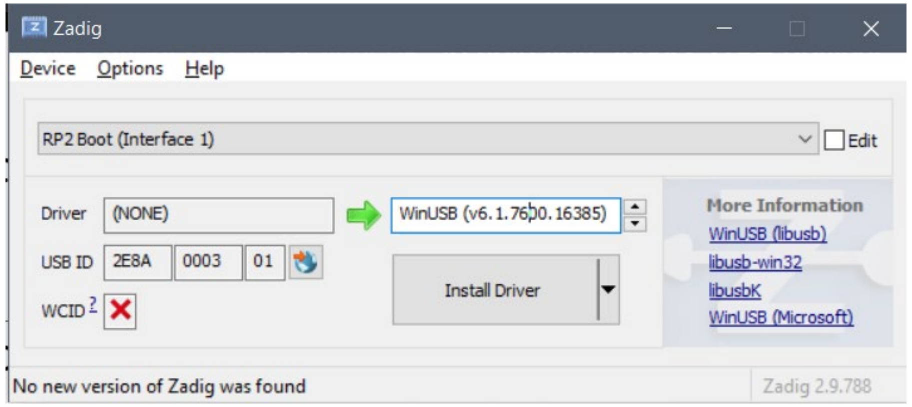
```
upload_port = COM10
```

But first: Fix windows

- Windows users can run into the following error message when uploading:

```
No new RPxxxx device found yet, waiting..
Warning: Picotool did not detect any RPxxxx devices in BOOTSEL mode. Upload might fail.
Uploading .pio\build\seeed_xiao_rp2040\firmware.elf
No accessible RP2040/RP2350 devices in BOOTSEL mode were found.
but:
Device at bus 1, address 24 appears to be a RP2040 device in BOOTSEL mode, but picotool was unable to connect. You may need to install a
driver via Zadig. See "Getting started with Raspberry Pi Pico" for more information
```

- Solution: Download zadig at https://zadig.akeo.ie/
- Run zadig and install winUSB driver



RP2 Boot (Interface 1)

Driver (NONE) → WinUSB (v6.1.7600.16385)

USB ID 2E8A 0003 01

WCID ✗

No new version of Zadig was found

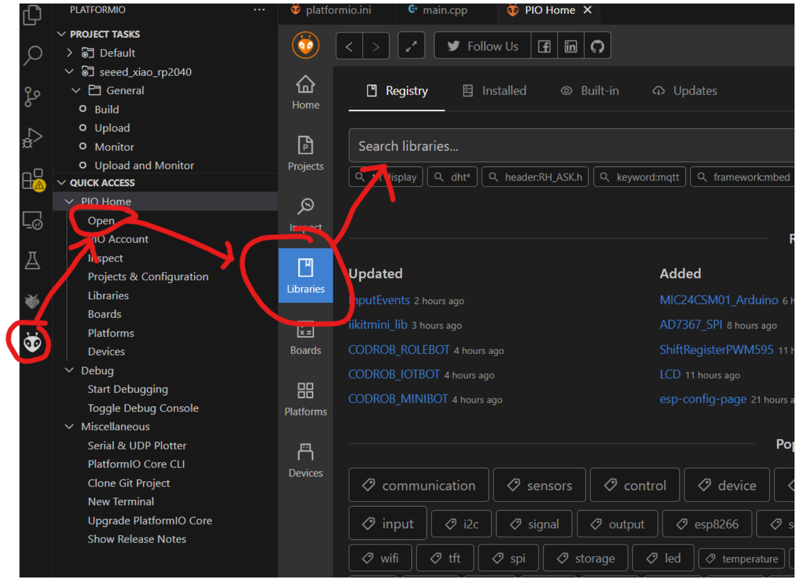
Zadig 2.9.788

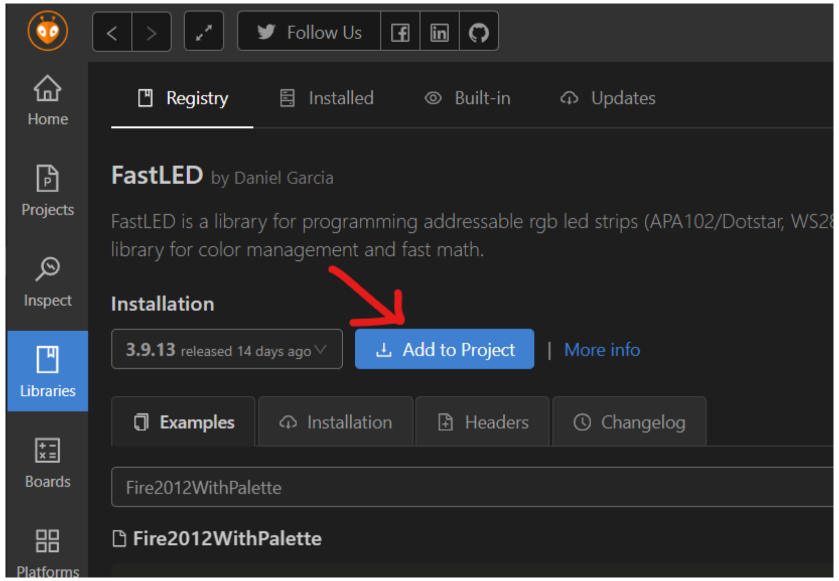
Blink 3 LEDs

```
#include <Arduino.h>

void setup() {
  // put your setup code here, to run once:
  pinMode(PIN_LED_R, OUTPUT);
  pinMode(PIN_LED_G, OUTPUT);
  pinMode(PIN_LED_B, OUTPUT);
  digitalWrite(PIN_LED_R, HIGH);
  digitalWrite(PIN_LED_G, HIGH);
  digitalWrite(PIN_LED_B, HIGH);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(PIN_LED_R, LOW);
  delay(200);
  digitalWrite(PIN_LED_R, HIGH);
  delay(200);
  digitalWrite(PIN_LED_G, LOW);
  delay(200);
  digitalWrite(PIN_LED_G, HIGH);
  delay(200);
  digitalWrite(PIN_LED_B, LOW);
  delay(200);
  digitalWrite(PIN_LED_B, HIGH);
  delay(200);
}
```

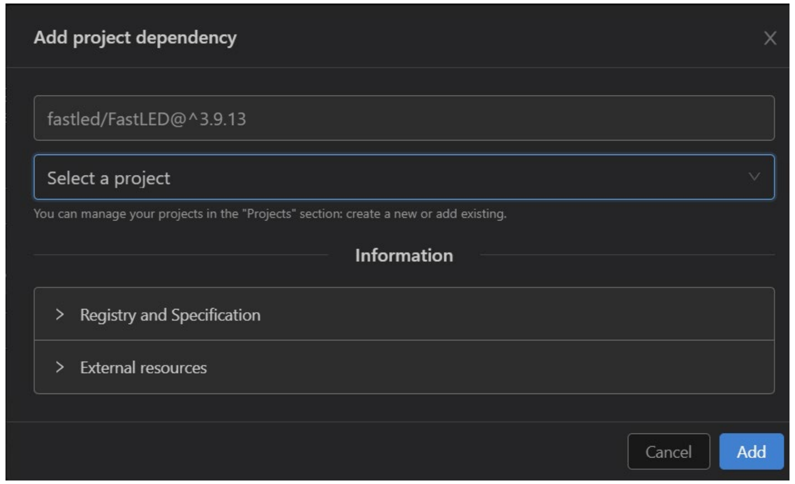
```
// XIAO RP2040 pins:

// pin 4 (D4) = I2C SDA
// pin 5 (D5) = I2C SCL
// pin 6 (D6) = UART TX
// pin 7 (D7) = UART RX / SPI Chip select
// pin 8 (D8) = SPI SCK
// pin 9 (D9) = SPI MISO
// pin 10 (D10) = SPI MOSI
// pin 11 = enable pin of RGB LED
//          (high = enabled)
// pin 12 = WS2812 data pin
// pin 16 = onboard green LED (use PIN_LED_G)
// pin 17 = onboard red LED (use PIN_LED_R)
// pin 25 = onboard blue LED (use PIN_LED_B)

// for ONBOARD LEDs: HIGH = OFF, LOW = ON
```

Add a library





FastLED is a library for programming addressable rgb led strips (APA102/Dotstar, WS28
library for color management and fast math.



fastled/FastLED@^3.9.13

Select a project

You can manage your projects in the "Projects" section: create a new or add existing.

- platformio.ini now contains the lib:

```
lib_deps = fastled/FastLED@^3.9.13
```

Blink the RGB LED

```
#include <Arduino.h>
#include <FastLED.h>
#define RGB_LED_ENABLE 11
#define RGB_DATA_PIN 12
#define NUM_LEDS 1

// Define the array of leds
CRGB leds[NUM_LEDS];

void setup() {
  pinMode(PIN_LED_R, OUTPUT);
  pinMode(PIN_LED_G, OUTPUT);
  pinMode(PIN_LED_B, OUTPUT);
  digitalWrite(PIN_LED_R, HIGH);
  digitalWrite(PIN_LED_G, HIGH);
  digitalWrite(PIN_LED_B, HIGH);
  pinMode(RGB_LED_ENABLE, OUTPUT);
  digitalWrite(RGB_LED_ENABLE, HIGH); // enable RGB LED
  FastLED.addLeds<NEOPIXEL, RGB_DATA_PIN>(leds, NUM_LEDS);
}
```

```
void loop() {
  leds[0] = CRGB::Red;
  FastLED.show();
  delay(200);
  leds[0] = CRGB::Black;
  FastLED.show();
  delay(200);
  leds[0] = CRGB::Green;
  FastLED.show();
  delay(200);
  leds[0] = CRGB::Black;
  FastLED.show();
  delay(200);
  leds[0] = CRGB::Blue;
  FastLED.show();
  delay(200);
  leds[0] = CRGB::Black;
  FastLED.show();
  delay(200);
}
```

Common pitfalls

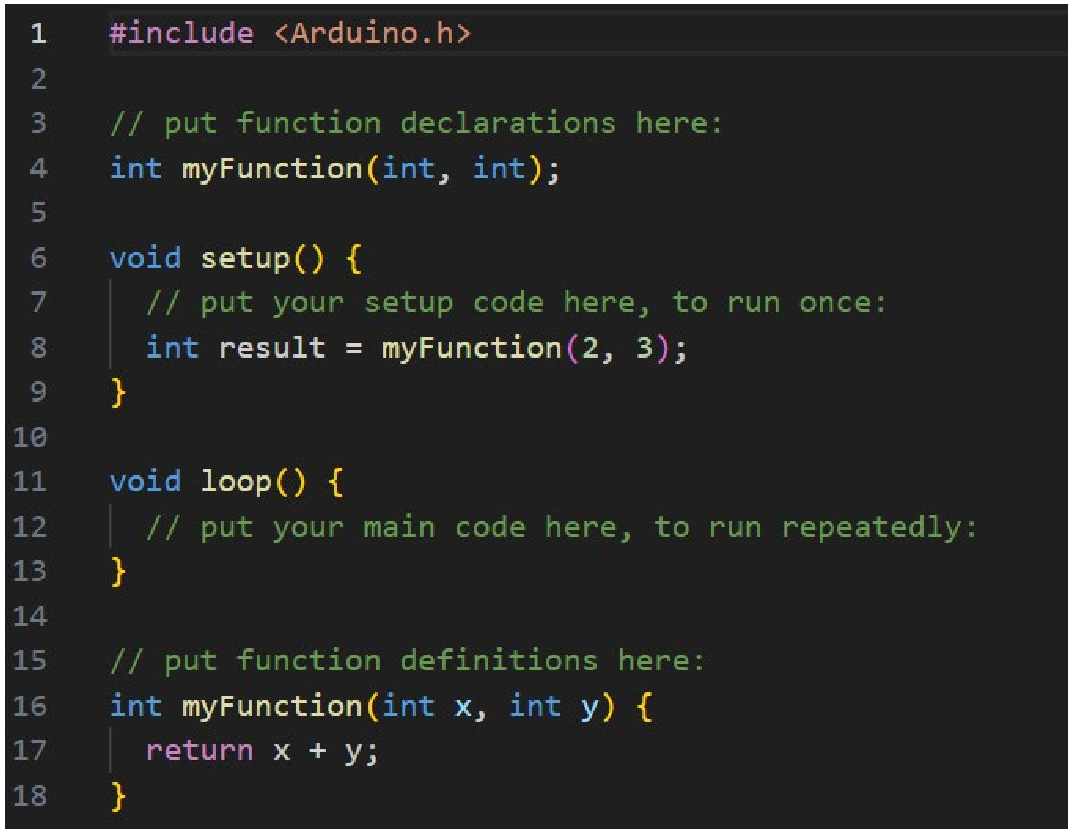
- If using Arduino framework, main.cpp should start with

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

- In Arduino IDE this is done automagically and is hidden from the user.

Common pitfalls

- Forward declaration of functions is required!
- Arduino IDE does this automatically for you. PlatformIO doesn't.



```
#include <Arduino.h>

// put function declarations here:
int myFunction(int, int);

void setup() {
  // put your setup code here, to run once:
  int result = myFunction(2, 3);
}

void loop() {
  // put your main code here, to run repeatedly:
}

// put function definitions here:
int myFunction(int x, int y) {
  return x + y;
}
```

Rule of thumb: copy the first line of the function definition and add a semicolon after it.

void myFunction();
void myFunctionWithParameters(int a, int b);

Further reading

- https://docs.platformio.org/en/latest/integration/ide/vscode.html#installation
- https://docs.platformio.org/en/latest/core/index.html
- https://docs.platformio.org/en/latest/projectconf/index.html