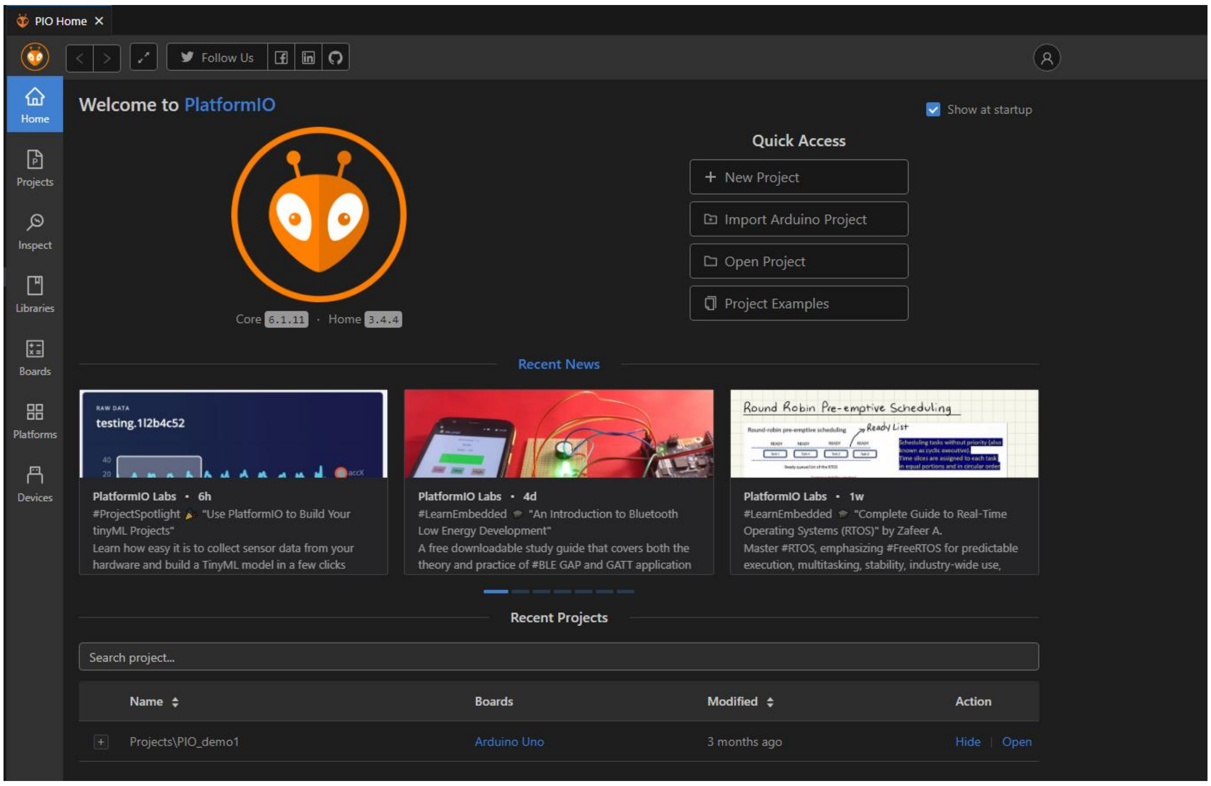
PlatformIO Workflow

Date : 20231226
Author : Wim Cranen (WCCandM)
Version : v0.1 – concept

Inhoudsopgave

1. Create Project.

Start VS Code and wait until the PlatformIO Home tab shows up.



PlatformIO Labs • 6h
#ProjectSpotlight "Use PlatformIO to Build Your tinyML Projects"
Learn how easy it is to collect sensor data from your hardware and build a TinyML model in a few clicks

PlatformIO Labs • 4d
#LearnEmbedded "An Introduction to Bluetooth Low Energy Development"
A free downloadable study guide that covers both the theory and practice of #BLE GAP and GATT application

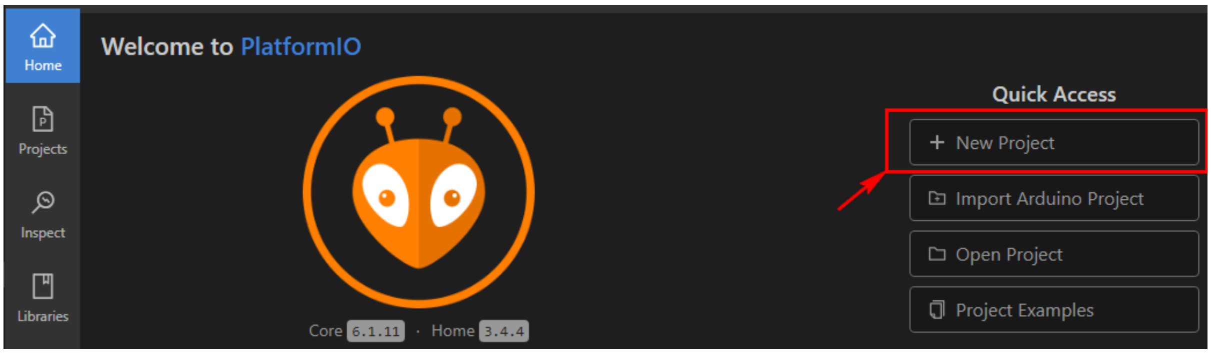
PlatformIO Labs • 1w
#LearnEmbedded "Complete Guide to Real-Time Operating Systems (RTOS)" by Zafeer A.
Master #RTOS, emphasizing #FreeRTOS for predictable execution, multitasking, stability, industry-wide use,

Recent Projects

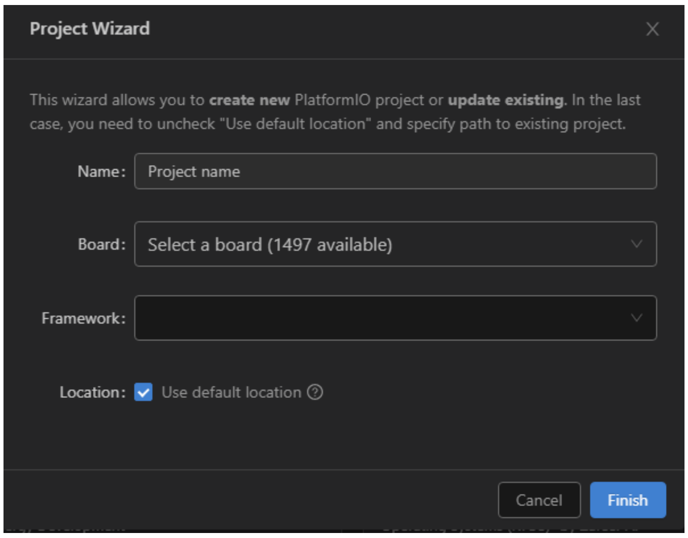
Search project...

Name	Boards	Modified	Action
Projects\PIO_demo1	Arduino Uno	3 months ago	Hide | Open

Go to PlatformIO home screen and click the New Project button.



The project wizard starts.



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

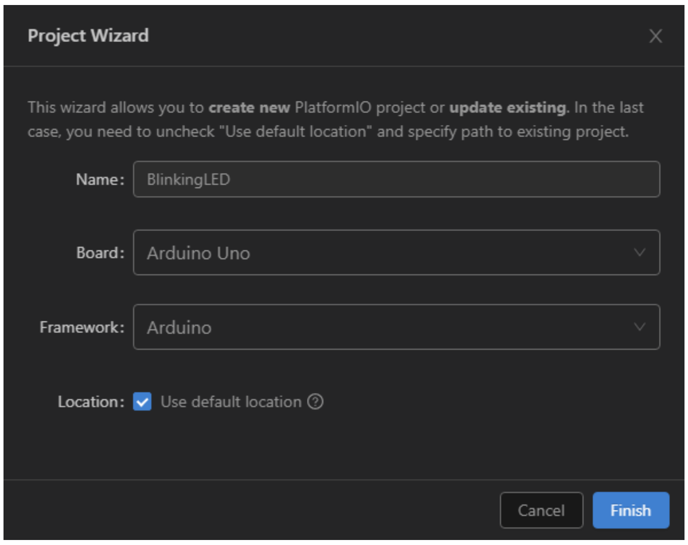
Board: Select a board (1497 available)

Framework:

Location: ✓ Use default location

Cancel Finish

Let us create an Project with the name “BlinkingLED” for an Arduino Uno.
The framework will be Arduino, Have the default location checked.
Then click the Finish button.



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.

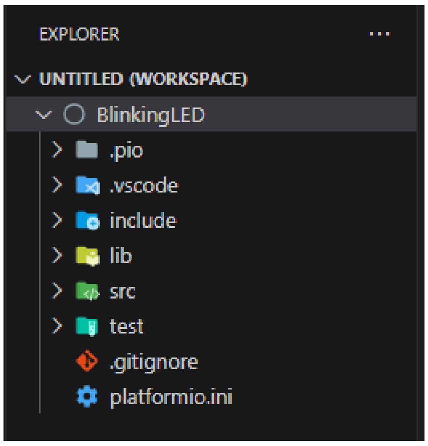
Name: BlinkingLED

Board: Arduino Uno

Framework: Arduino

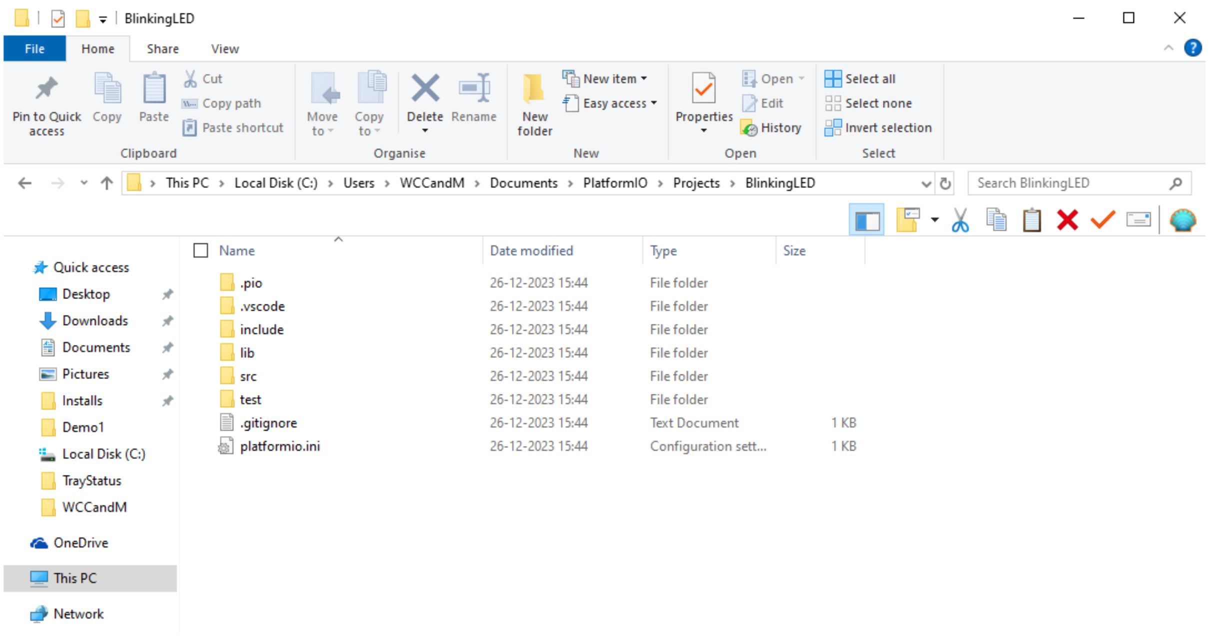
Location: [✓] Use default location

Wait until the Project Explorer shows the project and PlatformIO is ready with setting up your system for this project.



```
EXPLORER
UNTITLED (WORKSPACE)
  BlinkingLED
    .pio
    .vscode
    include
    lib
    src
    test
    .gitignore
    platformio.ini
```

This is the where the project resides on my system.

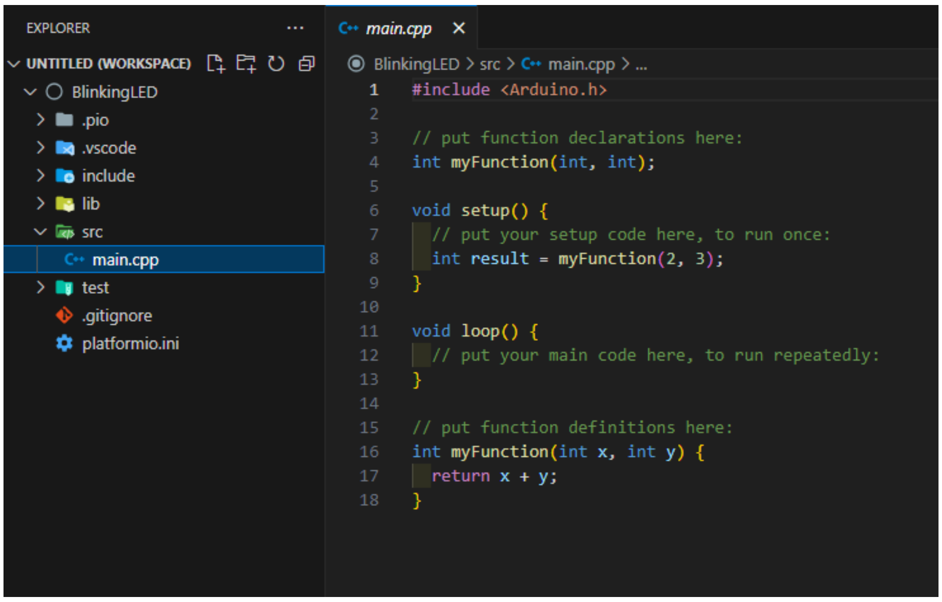


Name	Date modified	Type	Size
.pio	26-12-2023 15:44	File folder	
.vscode	26-12-2023 15:44	File folder	
include	26-12-2023 15:44	File folder	
lib	26-12-2023 15:44	File folder	
src	26-12-2023 15:44	File folder	
test	26-12-2023 15:44	File folder	
.gitignore	26-12-2023 15:44	Text Document	1 KB
platformio.ini	26-12-2023 15:44	Configuration sett...	1 KB

The .ino sketch will have to go into the /src/ (source) directory, in the mail.cpp file.

In the project explorer open this file and start editing.

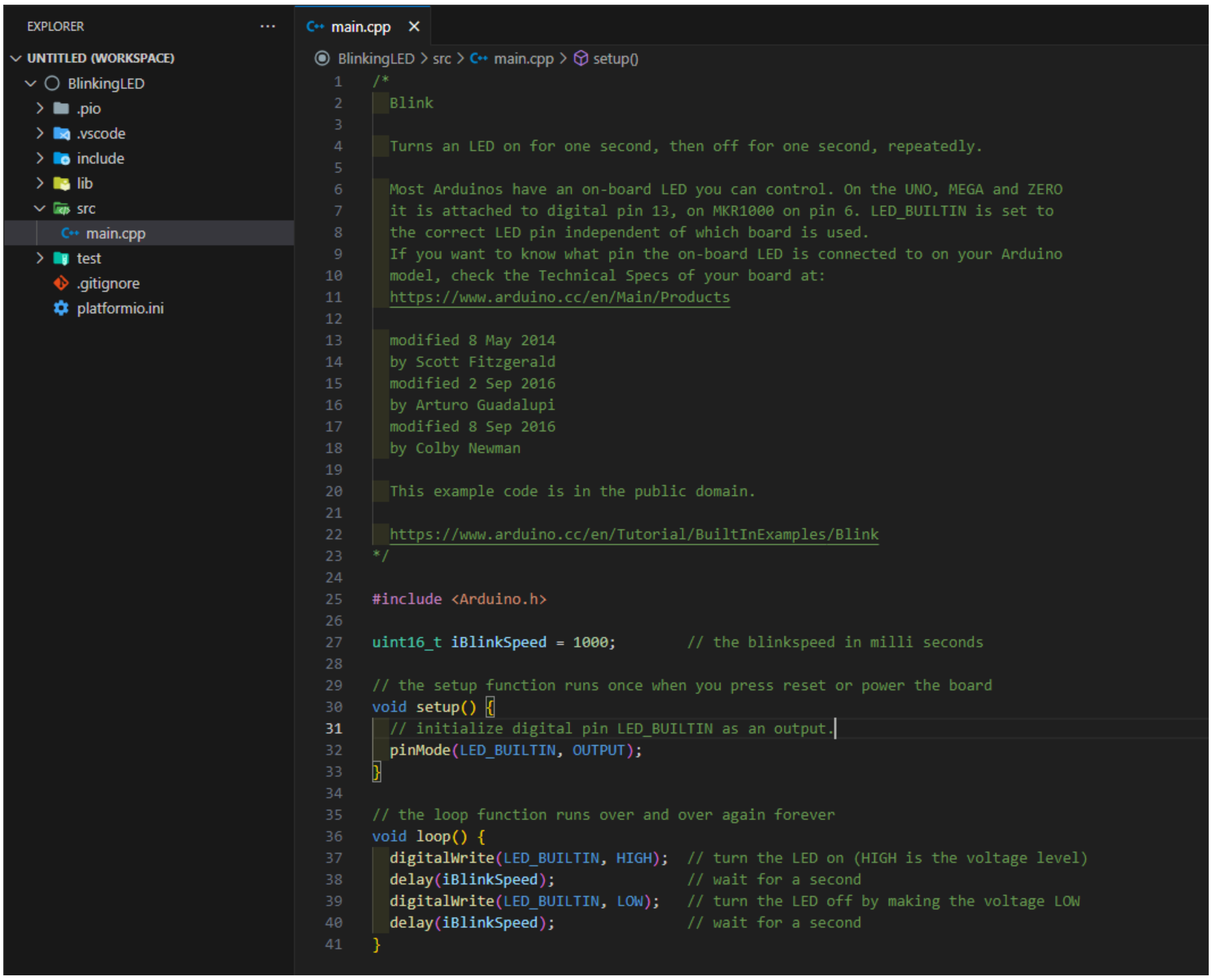
2. Enter code.



```cpp
#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;
}
```

Have a look at the upper line in the code. `#include <Arduino.h>` must always be part of the project (if it is an Arduino project).

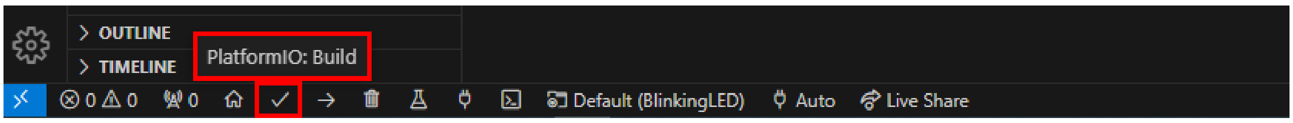
Now copy the code of the `Blink.ino` project in this window.
Then the window looks like below.



```cpp
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

#include <Arduino.h>

uint16_t iBlinkSpeed = 1000;        // the blinkspeed in milli seconds

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(iBlinkSpeed);               // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(iBlinkSpeed);               // wait for a second
}
```

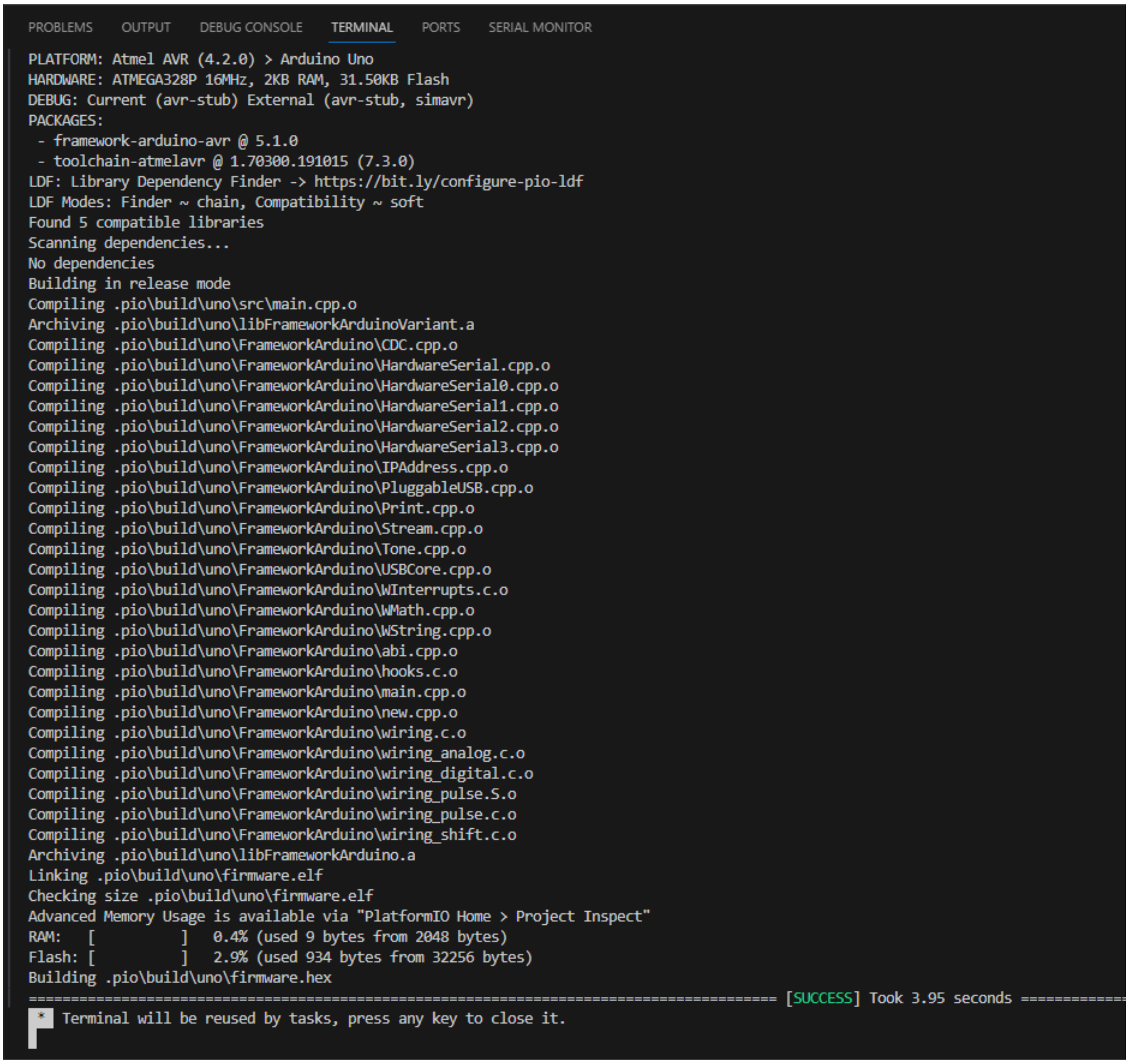
3. Build the Project.

Now we can compile (Build) the project to see if everything is ok.
Click on this symbol (check) in the lower toolbar.



Then the project starts building and if you enlarge the Terminal window, you will see the progress of the compiler, building the project.

At the end you will see something like this if everything was ok.
If not, solve the errors.



```
PLATFORM: Atmel AVR (4.2.0) > Arduino Uno
HARDWARE: ATMEGA328P 16MHz, 2KB RAM, 31.50KB Flash
DEBUG: Current (avr-stub) External (avr-stub, simavr)
PACKAGES:
 - framework-arduino-avr @ 5.1.0
 - toolchain-atmelavr @ 1.70300.191015 (7.3.0)
LDF: Library Dependency Finder -> https://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 5 compatible libraries
Scanning dependencies...
No dependencies
Building in release mode
Compiling .pio\build\uno\src\main.cpp.o
Archiving .pio\build\uno\libFrameworkArduinoVariant.a
Compiling .pio\build\uno\FrameworkArduino\CDC.cpp.o
Compiling .pio\build\uno\FrameworkArduino\HardwareSerial.cpp.o
Compiling .pio\build\uno\FrameworkArduino\HardwareSerial0.cpp.o
Compiling .pio\build\uno\FrameworkArduino\HardwareSerial1.cpp.o
Compiling .pio\build\uno\FrameworkArduino\HardwareSerial2.cpp.o
Compiling .pio\build\uno\FrameworkArduino\HardwareSerial3.cpp.o
Compiling .pio\build\uno\FrameworkArduino\IPAddress.cpp.o
Compiling .pio\build\uno\FrameworkArduino\PluggableUSB.cpp.o
Compiling .pio\build\uno\FrameworkArduino\Print.cpp.o
Compiling .pio\build\uno\FrameworkArduino\Stream.cpp.o
Compiling .pio\build\uno\FrameworkArduino\Tone.cpp.o
Compiling .pio\build\uno\FrameworkArduino\USBCore.cpp.o
Compiling .pio\build\uno\FrameworkArduino\WInterrupts.c.o
Compiling .pio\build\uno\FrameworkArduino\WMath.cpp.o
Compiling .pio\build\uno\FrameworkArduino\WString.cpp.o
Compiling .pio\build\uno\FrameworkArduino\abi.cpp.o
Compiling .pio\build\uno\FrameworkArduino\hooks.c.o
Compiling .pio\build\uno\FrameworkArduino\main.cpp.o
Compiling .pio\build\uno\FrameworkArduino\new.cpp.o
Compiling .pio\build\uno\FrameworkArduino\wiring.c.o
Compiling .pio\build\uno\FrameworkArduino\wiring_analog.c.o
Compiling .pio\build\uno\FrameworkArduino\wiring_digital.c.o
Compiling .pio\build\uno\FrameworkArduino\wiring_pulse.S.o
Compiling .pio\build\uno\FrameworkArduino\wiring_pulse.c.o
Compiling .pio\build\uno\FrameworkArduino\wiring_shift.c.o
Archiving .pio\build\uno\libFrameworkArduino.a
Linking .pio\build\uno\firmware.elf
Checking size .pio\build\uno\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [          ]   0.4% (used 9 bytes from 2048 bytes)
Flash: [          ]   2.9% (used 934 bytes from 32256 bytes)
Building .pio\build\uno\firmware.hex
======================================================================================= [SUCCESS] Took 3.95 seconds ===========
 *  Terminal will be reused by tasks, press any key to close it.
```

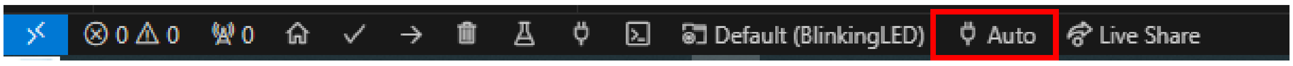
After building the project is ok, you can send the project to the specified board.

4. Load the project to the specified board.

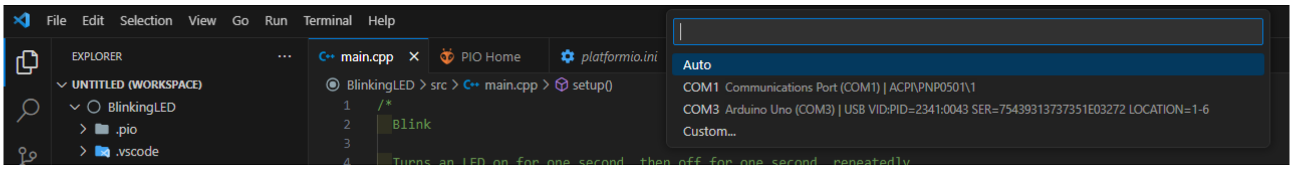
Connect the board. This board is connected through an USB cable type A-B.

If your system is in a virtual machine, do not forget to pass the serial port from the host machine to this VM.

In the lower toolbar, check if PlatformIO is in Auto mode. In the most cases, PlatformIO will select the correct port.



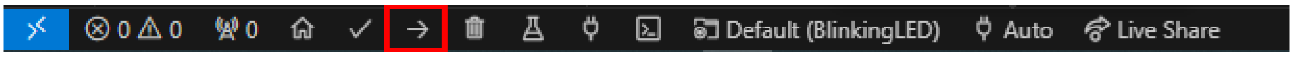
If you are unlucky and cannot establish a connection, click this button.
Then a windows opens at the top, Select the correct port.



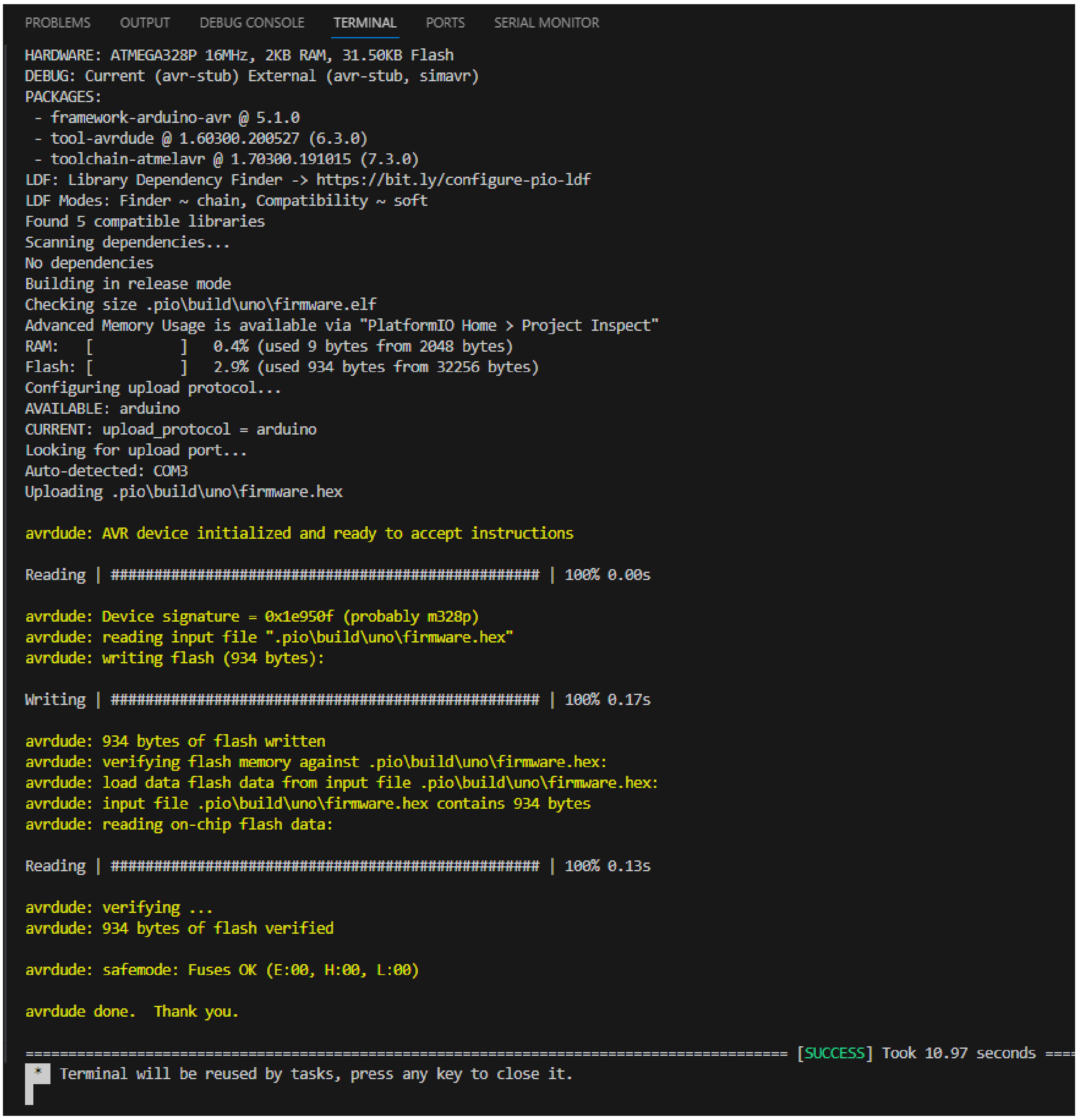
```
/*
 Blink

 Turns an LED on for one second, then off for one second, repeatedly.
```

```
Auto
COM1  Communications Port (COM1) | ACPI\PNP0501\1
COM3  Arduino Uno (COM3) | USB VID:PID=2341:0043 SER=75439313737351E03272 LOCATION=1-6
Custom...
```

If there is a connection, click the arrow in the lower toolbar to send the built program to the board.



If the program is correctly loaded, then your terminal output will show information.



```
HARDWARE: ATMEGA328P 16MHz, 2KB RAM, 31.50KB Flash
DEBUG: Current (avr-stub) External (avr-stub, simavr)
PACKAGES:
 - framework-arduino-avr @ 5.1.0
 - tool-avrdude @ 1.60300.200527 (6.3.0)
 - toolchain-atmelavr @ 1.70300.191015 (7.3.0)
LDF: Library Dependency Finder -> https://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 5 compatible libraries
Scanning dependencies...
No dependencies
Building in release mode
Checking size .pio\build\uno\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [          ]   0.4% (used 9 bytes from 2048 bytes)
Flash: [          ]   2.9% (used 934 bytes from 32256 bytes)
Configuring upload protocol...
AVAILABLE: arduino
CURRENT: upload_protocol = arduino
Looking for upload port...
Auto-detected: COM3
Uploading .pio\build\uno\firmware.hex

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: reading input file ".pio\build\uno\firmware.hex"
avrdude: writing flash (934 bytes):

Writing | ################################################## | 100% 0.17s

avrdude: 934 bytes of flash written
avrdude: verifying flash memory against .pio\build\uno\firmware.hex:
avrdude: load data flash data from input file .pio\build\uno\firmware.hex:
avrdude: input file .pio\build\uno\firmware.hex contains 934 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.13s

avrdude: verifying ...
avrdude: 934 bytes of flash verified

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

avrdude done.  Thank you.

========================================================================================== [SUCCESS] Took 10.97 seconds ====
 *  Terminal will be reused by tasks, press any key to close it.
```