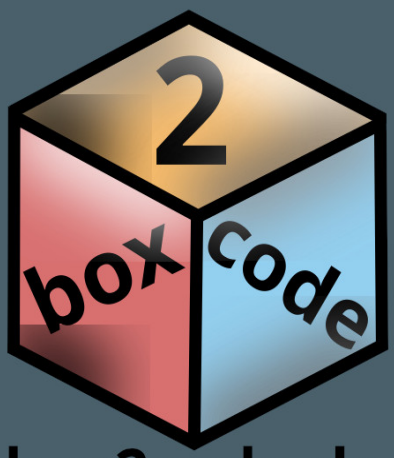


box2code.de

Embedded Spaß mit gutem tooling

Stefan Scholz - 2022

Inhaltsverzeichnis

Embedded Spaß mit gutem tooling 1.1
Installation 1.2
Projekt anlegen 1.3
Ausprobieren 1.4

Einleitung

Dieser Artikel richtet sich an Bastler, die schon erste Schritte mit Arduino oder mbed unternommen haben, aber mit den Möglichkeiten der dafür standardmäßig angebotenen IDEs etwas eingeschränkt arbeiten.

Warum dieser Artikel

Ich habe Freunde und Bekannte, die schon etwas mit Mikrocontrollern experimentiert haben und die Möglichkeiten einer guten IDE garnicht kannten. Früher war es sehr aufwendig sich eine gute Toolchain für einen Controller aufzusetzen. Durch VSCode und PlatformIO ist das aber heute ein Kinderspiel und man muss auf Nichts verzichten.

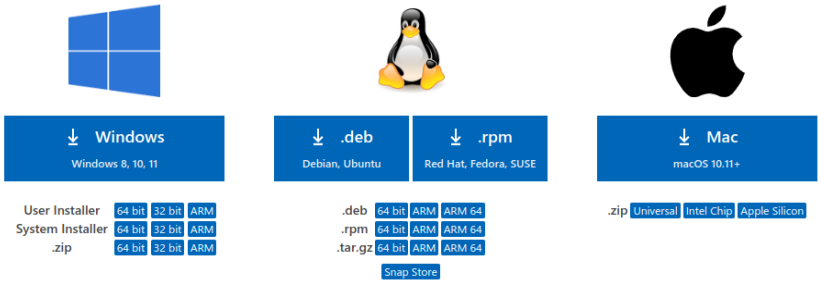
Ist VSCode Visual Studio ?

Nein, die Initiative zu VSCode kommt zwar von Microsoft. Doch ist VSCode eine völlige Neuentwicklung einer IDE, die auf Electron basiert. Im Grunde ist Electron ein Chrome-ähnlicher Browser und VSCode eine Webanwendung (html, javascript) darin. Wer das nicht glauben kann, sollte mal den link https://vscode.dev/ in seinem Browser öffnen. Dann läuft die Anwendung VSCode im Browserfenster. Natürlich nur eingeschrängt, da Browser wie Chrome oder Firefox Zugriffe auf das System (z.B. Dateisystem) grundsätlich verbieten. Trotzdem - schöne Spielerei :-) Da Electron für alle gängigen Desktop-Plattformen verfügbar ist läuft VSCode auf jeder Platform nahezu identisch. Ich arbeite mit Linux, doch du kannst Alles in diesem Artikel genauso mit deinem Windows-PC machen.

Installation

Von VSCode

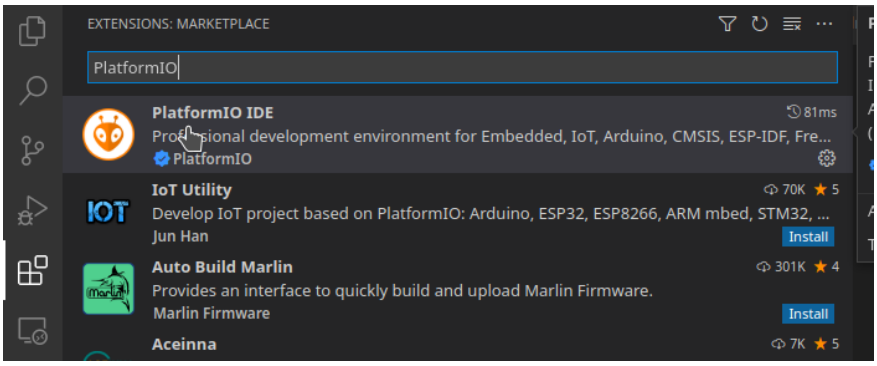
Gehe auf https://code.visualstudio.com/download und hole dir den Installer passend zu deiner Platform:



Der Installer ist sehr komfortabel und du solltest keine Probleme beim Installieren haben.

Vom PlatformIO-Plugin

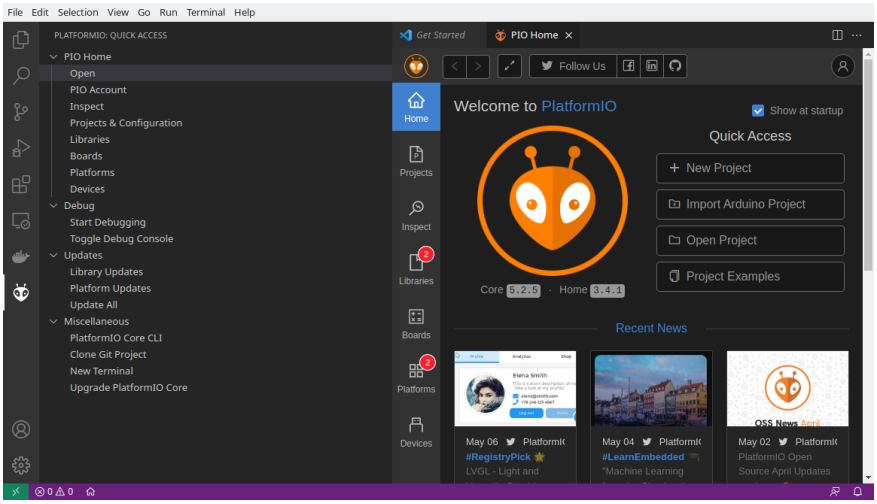
Nachdem du VSCode gestartet hast, siehst du links am Rand ein Icon mit 4 kleinen Kästchen. Damit kannst du nach Plugins suchen und diese installieren:



Installiere PlatformIO und warte etwas (manchmal auch etwas länger :-)) Am linken Rand von VSCode erscheint nun das Symbol von PlatformIO:

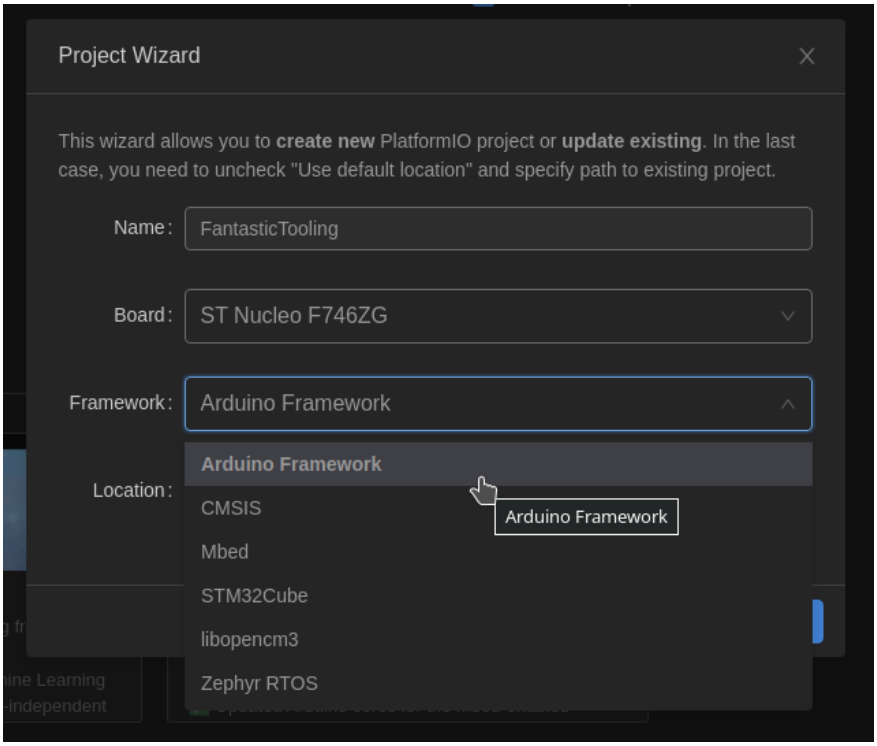
Erstes Projekt anlegen

Klicke nun auf das neue PlatformIO-Icon - dann auf Open - dann auf "New Project"



Board und Framework wählen

Hier kannst du einen Projektnamen eingeben, dein Board mit dem du arbeitest auswählen und natürlich das Framework, dass du verwenden willst:



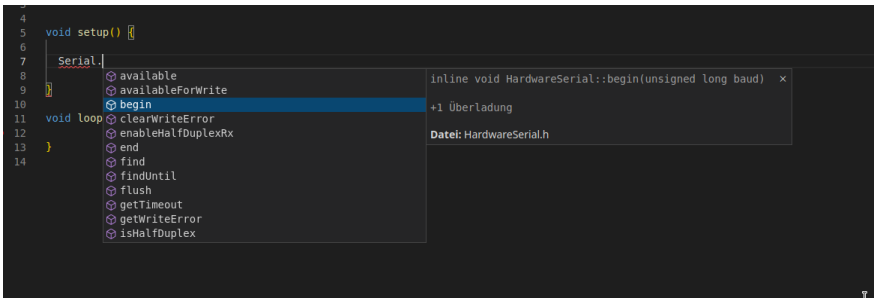
Ich hab mich für ein Nucleo-Board von ST entschieden und erstmal für Arduino, da das, glaube ich, für die Meisten von interesse ist. Ich mag übrigens mbed sehr gerne - vorallem für die Controller von ST.

PlatformIO legt das neue Projekt als Verzeichnis unter ~/Dokumente/PlatformIO ab. Eigentlich ganz nett.

Ausprobieren

Code completition

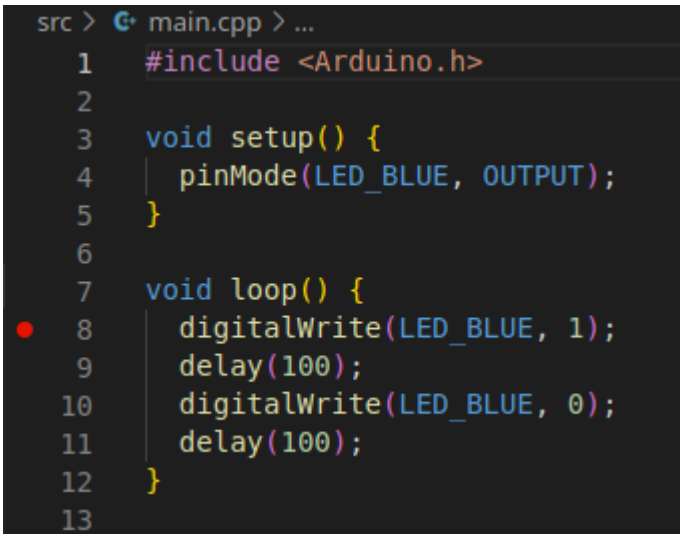
PlatformIO hat uns eine standard main.cpp mit dem für Arduino typischen Inhalt angelegt:



Gib einfach mal Serial und danach einen Punkt ein. Es öffnet sich ein Drop-Down-Menü mit Allem was du mit diesem Objekt anstellen kannst. Das ist ein Feature, dass gerade Benutzer der Arduino-IDE begeistern sollte. Nicht mehr in der Doku nachlesen, sondern Alles beim implementieren parat haben ist was tolles :-)

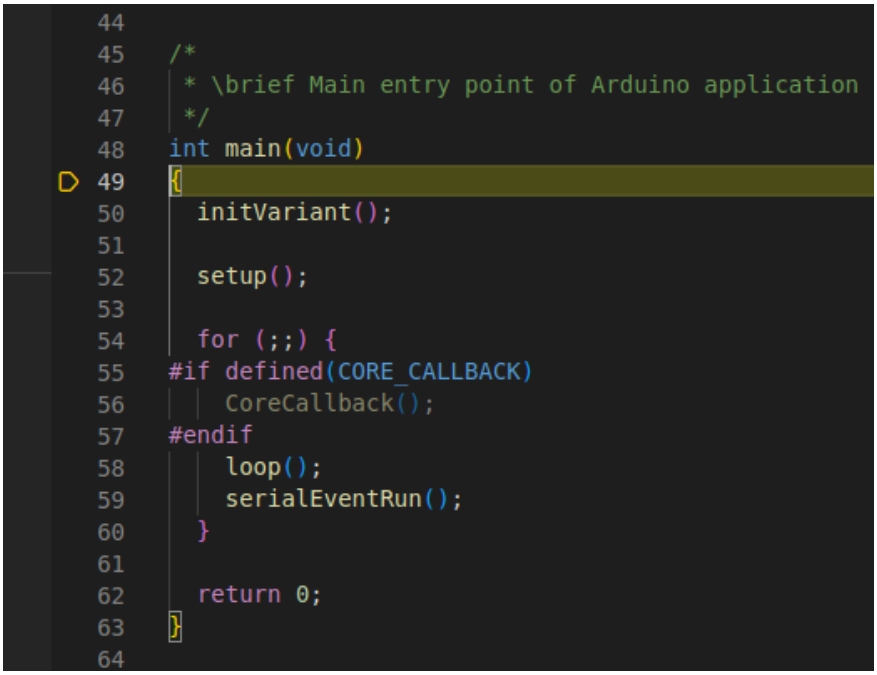
Debuggen

Fülle deine main.cpp etwas mit code. Was genau du dort machst spielt keine große Rolle. Ich hab einfach mal eine LED blinken lassen:



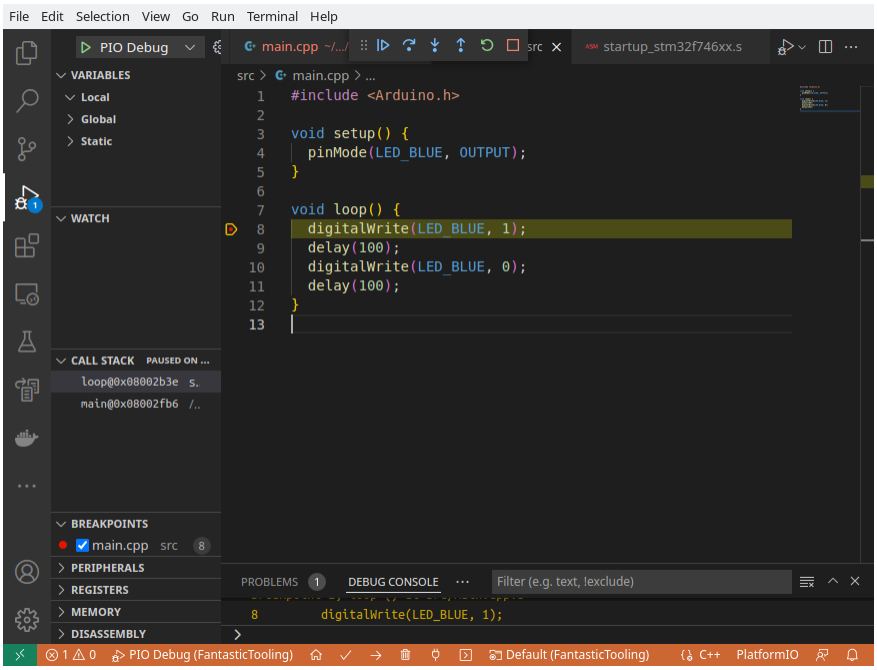
Jetzt setze vor eine Zeile die dich interessiert durch klicken neben der Zeilennummer einen kleinen roten Punkt (Breakpoint). Dein Programm wird später an dieser Stelle anhalten.

Mit main.cpp geöffnet drückst du jetzt einfach mal F5 (Alternativ Menü Run->Debug). Deine Software wird jetzt gebaut. Anschließend bleibt der Debugger an der ersten Zeile deines Programms stehen. Das ist nicht etwa setup(), wie du vielleicht denkst. Sondern in der echten main()-Funktion die Arduino für dich angelegt hat:



Daran siehst du, dass Arduino eigentlich ganz normales C++ ist. Drücke einfach nochmals F5 (oder den Continue-Pfeil oben in der Mitte)

Dein Programm hält jetzt an der Stelle mit deinem roten Punkt an:



Mit F10 kannst du jetzt dein Programm einen einzelnen Schritt machen lassen. Ist an der Haltestelle ein Funktionsaufruf, so kannst du mit F11 in diese Funktion "hineinspringen".

Wenn du schon mit einem Debugger gearbeitet hast, werde ich dich mittlerweile langweilen. Falls nicht hast du hier noch einiges zum Ausprobieren - aber das würde den Rahmen sprengen :-)

Fairerweise muss ich noch erwähnen, dass nicht alle Boards einen eingebauten Debugger mitbringen. Findet man bei ST auf jedem Board einen, so bringen z.B. ATMegas und ESPs keine Debug-Peripherie mit.

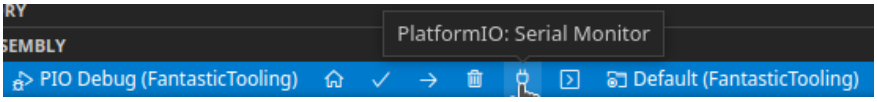
Es gibt aber grundsätzlich für jede MCU, die ich kenne, einen Debug-Adapter mit dem PlatformIO auch umgehen kann. Preise und Aufwand sind aber sehr unterschiedlich.

Terminal

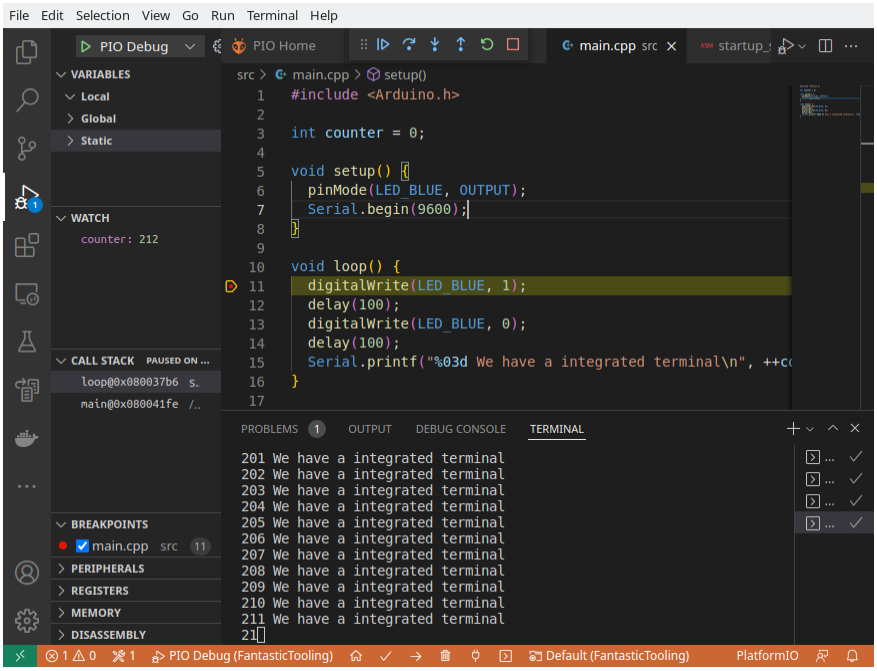
Gerade bei den MCUs ohne eingebauten Debugger ist ein Terminal von großer Wichtigkeit, damit man sich ab und zu mal ausgeben kann was das Programm gerade tut.

Glücklicherweise bringt PlatformIO gleich eines mit.

Kommt zwar ein wenig spät ist aber dennoch sehr wichtig - PlatformIO hat am unteren VSCode-Bildschirmrand noch eine Toolbar, die oft gebraucht wird.



Mit dem Pfeil nach rechts spielst du z.B. dein Programm in eine MCU die nicht über einen Debug-Adapter angeschlossen ist. Gehe einfach mal mit der Maus über alle Symbole dort - ist sehr selbsterklärend, was man damit machen kann. Unteranderem findest du dort auch das eingebaute Terminal:



Unkonfiguriert ist die Standardeinstellung erstmal 9600 8N1 und es kann sein, dass du gefragt wirst, welchen Adapter du verwenden willst.