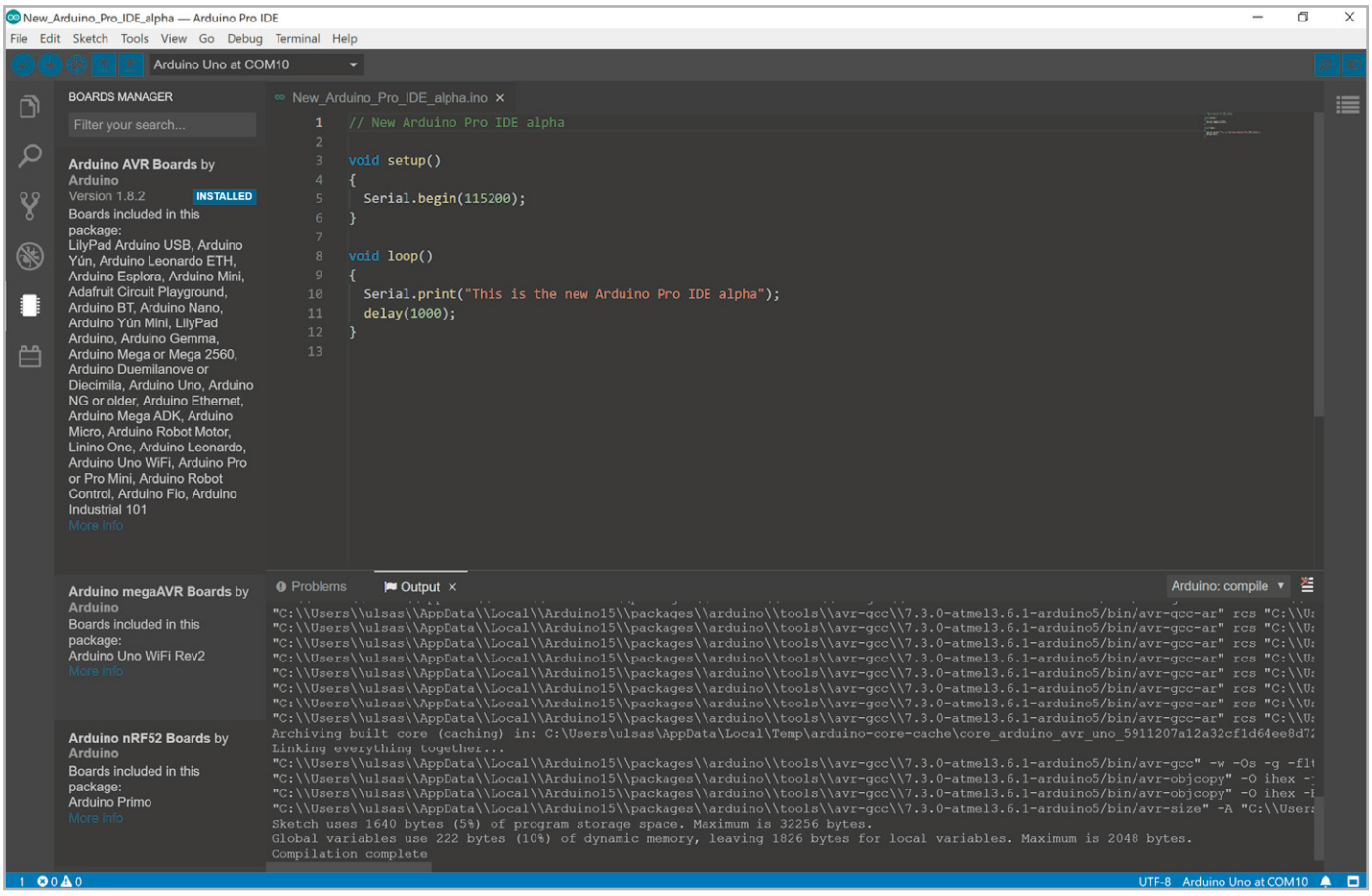
Arduino wird erwachsen

Programmierumgebung Arduino Pro IDE mit Profi-Features

Die Einfachheit der Mikrocontroller-Boards und der Programmierumgebung hat Arduino in den letzten 15 Jahren zu einem der populärsten Entwicklungstools für Maker, Elektronik-Enthusiasten, Programmiereinsteiger, Künstler und Anwender aus vielen anderen Bereichen gemacht. Millionenfach dürften die quelloffenen Boards – ob original oder als Nachbau – zusammen mit der Arduino IDE als einfache, aber wirkungsvolle Plattform für Projekte aller Art mittlerweile eingesetzt worden sein. Seit letztem Herbst gibt es nun einen Ansatz für eine professionellere Programmierumgebung (IDE) aus dem Hause Arduino [1]. Wir haben uns die Software, die sich noch im Alpha-Stadium befindet, genauer angeschaut.



Aus gut mach besser

Die klassische Arduino IDE (IDE = Integrated Development Environment, deutsch: Programmierumgebung, Bild 1 und Bild 2), die momentan in der Version 1.8.12 vorliegt, besticht vor allem durch ihre Einfachheit. Für den Anwender sind Entwicklungsboards von den klassischen Arduinos wie dem UNO, über ARM-Entwicklungsboards wie den SAMD21-Varianten (mit 32-Bit ARM Cortex M0+) bis hin zu Boards von Drittanbietern wie den Espressif SoCs ESP8266 und ESP32 leicht einzubinden. Die Arduino-Oberfläche vereint alle diese doch in der Hardware recht unterschiedlichen Boards und lässt sie mit den gleichen Code-Funktionen nahezu identisch programmieren. Ein typisches Beispiel dafür ist der Blink-Sketch, als „Hello World"-Äquivalent der Software-Welt, der als erster Test bei neuer Hardware gilt und eine LED im Sekundentakt blinken lässt.

In den Jahren hat sich am Unterbau der Arduino IDE – vom normalen Nutzer wahrscheinlich eher unbemerkt – eine Menge getan, um neue Features wie das Einbinden von Boards über Boardverwalter-URLs oder das Suchen und Installieren von Programmbibliotheken zu ermöglichen.

Trotz der vielen hinzugefügten Funktionen blieb die Oberfläche vom Aussehen und der Benutzung her nahezu gleich. Sicherlich eines der Geheimnisse für den großen Erfolg des Systems Arduino.

Von Profis, die für das Prototyping nach anfänglichem Zögern immer häufiger die Arduino-Plattform, sowohl was die Hardware als auch die Programmierumgebung angeht, genutzt haben, kam allerdings auch Kritik an fehlenden Eigenschaften. Vermisst wurde vor allem die automatische Vervollständigung bei der Bearbeitung von Quellcode, die Möglichkeit zum Debugging oder die Integration in Systeme zur verteilten Versionsverwaltung wie beispielsweise Git [2]. Als Alternativen boten sich für diesen Personenkreis bisher Atmel Studio [3] oder PlatformIO IDE [4] bzw. andere professionelle Entwicklungsumgebungen an.

Eine Menge neue Features

Mit der neuen Arduino Pro IDE plant man nun einen grundlegenden Wandel, der vorwiegend im Untergrund schlummert. Das Stichwort heißt Arduino CLI (Command Line Interface, deutsch: Kommandozeile). Hierzu später noch etwas mehr. Doch bevor wir uns die Details anschauen und einen Blick auf und unter die Oberfläche der neuen Programmierumgebung in der Alpha-Version wagen, hier zunächst die von Arduino für die Pro IDE angekündigten Features:

- Moderne, umfassende Entwicklungsumgebung
- Dual Mode: klassische Arduino-Oberfläche oder professionelle Datei-/Panelansicht
- Dark Mode
- Möglichkeit für die Entwicklung großer, mehrdateienbasierter und Repository-gestützter Projekte
- Offen für Boards und Plug-ins von Drittanbietern
- Automatische Code-Vervollständigung
- Unterstützung für Arduino-, Python- und JavaScript-Code
- Neuer Board-Manager, Bibliothek-Manager und serieller Monitor
- Git-Integration
- Debugging: Debugger mit der Möglichkeit, Breakpoints zu setzen oder den Code schrittweise ablaufen zu lassen

Neues Back- und Frontend

Die geplanten Features sind ein großes Vorhaben, da einige bisher überhaupt noch nicht in der Programmierumgebung vorhanden waren. Zudem ist es zu diesem Zeitpunkt wichtig zu wissen, dass es sich um eine Alpha-Software in der Doppelnull-Version 0.0.6 handelt. Produktiv sollte man diese ersten Versionen, die auf GitHub zum Download zur Verfügung gestellt werden [5], also nicht einsetzen.

Doch was steckt nun hinter den ganzen Neuerungen? Im Backend arbeitet jetzt das Arduino CLI. Auf dieser Kommandozeile kann man zahlreiche Werkzeuge nutzen, beispielsweise den Board-/Bibliotheken-Manager, den Build-Prozess, das Uploaden oder Kompilieren oder das Erstellen von Sketchen. Damit ist gleichzeitig die Möglichkeit gegeben, die Arduino-Unterstützung auch in anderen Plattformen und Editoren wie VSCode, Eclipse oder Atom zu nutzen.

Für die webbasierte Plattform Arduino Create wird das Arduino CLI bereits genutzt. Durch die Unterstützung für einen Daemon-Mode und das gRPC-Interface (gRPC ist ein Protokoll zum Aufruf von Funktionen in verteilten Computersystemen) können die Ausführungsschicht und das User-Interface, also z. B. die Arduino Pro IDE, auch auf unterschiedlichen Maschinen laufen.

Momentan arbeitet Arduino daran, alle herkömmlichen Produkte auf die Basis von Arduino CLI zu stellen. Mehr zum Arduino CLI gibt es in einem Blogbeitrag von Arduino [6].

Auch das Frontend und damit die Oberfläche haben einen neuen technischen Unterbau bekommen. Arduino nutzt hier das Open-Source-IDE Theia Framework [7] von Eclipse Foundation. Theia wiederum

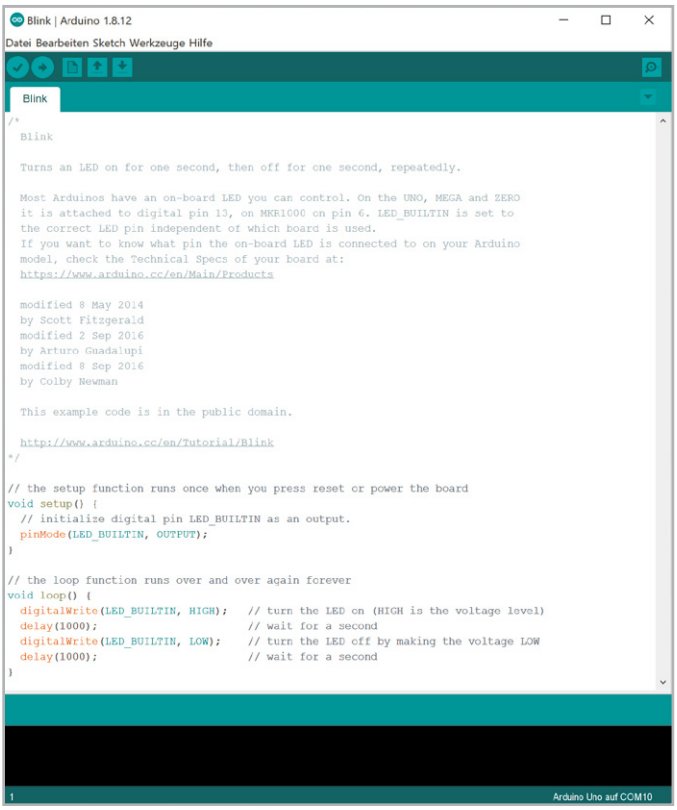


Bild 1: Die klassische Arduino IDE aus 2020 (Version 1.8.12) …

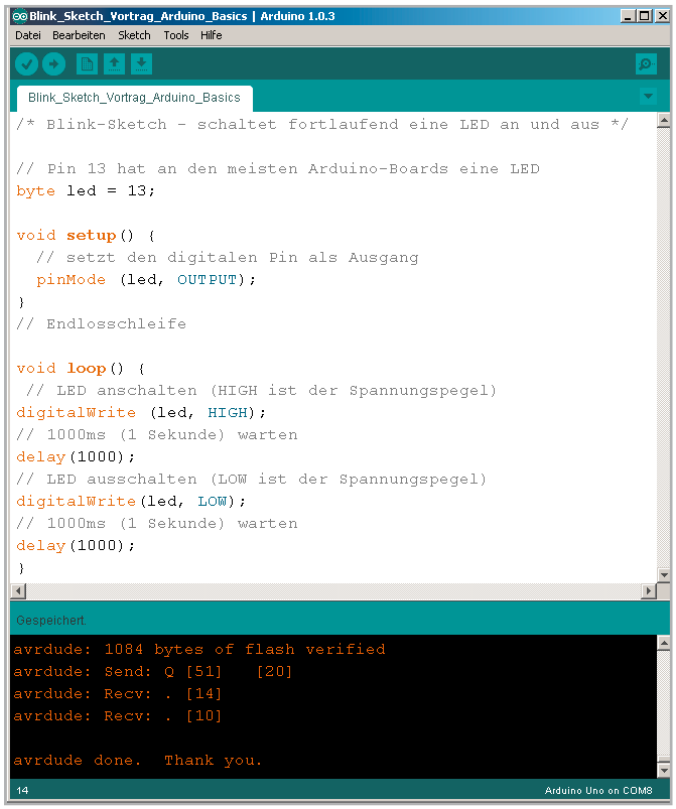


Bild 2: … und im Vergleich dazu aus 2013 (Version 1.0.3)

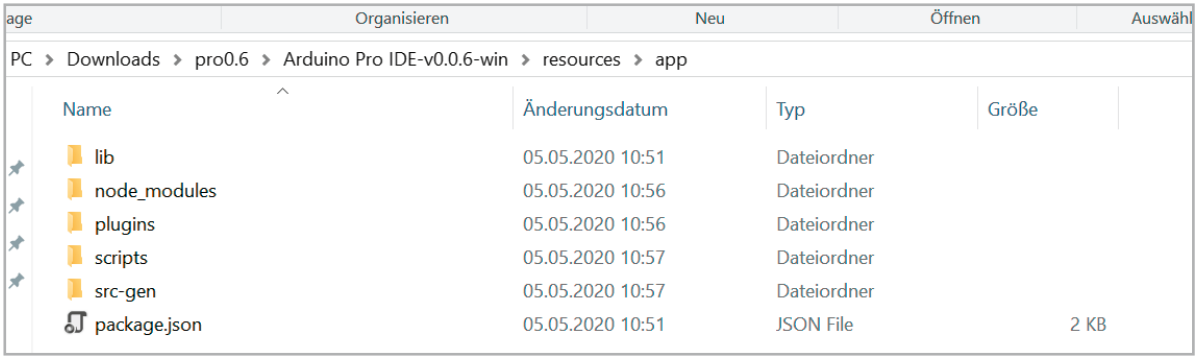


Bild 3: Die Arduino Pro IDE nutzt im Unterbau JavaScript und Node-Module.

basiert auf Electron [8], das durch die Nutzung von Webtechnologien wie JavaScript, HTML und CSS plattformübergreifende Desktop-Anwendungen zur Verfügung stellen kann. Schaut man in die Ordnerstruktur der Alpha-Version, fällt hier auch gleich die Verwendung von JavaScript bzw. Node-Modulen auf (Bild 3).

Features im Einzeltest

Zunächst muss man sich die je nach Betriebssystem – zurzeit gibt es Versionen für Windows, Mac OS X und Linux64 – benötigten Dateien für die Arduino Pro IDE unter [5] herunterladen und entpacken. Wir haben die Version unter Windows getestet. Dort schreitet der Microsoft-Defender SmartScreen ein und warnt vor der Verwendung des Programms. Hier sollte man selbst entscheiden, ob man das Programm auf einem produktiven System oder wie wir in einem eigens dafür aufgesetzten Testsystem startet.

Nach dem Start fällt zunächst die in verschiedene Panels aufgeteilte Programmierumgebung im Vergleich zu der klassischen Arduino-IDE Oberfläche und dem Dark Theme als vorausgewähltem Editor-Style auf. Diesen kann man unter File → Settings → Color Theme ändern.

Mit dem Schiebeschalter rechts oben in (Bild 4: 6) bei kann man zwischen normalem und Advanced Mode umschalten, was wir als Erstes tun. In Bild 4 schauen wir uns die Oberfläche mit den verschiedenen Panels, Menüs und Icons einmal näher an.

Die Oberfläche

Schauen wir uns im Einzelnen die Elemente der neuen Oberfläche an:

1. Das klassische Arduino IDE Menü. Neu sind View, Go, Debug und Terminal.
2. Hier fällt das kleine Käfer-Symbol auf, welches das Debugging startet. Außerdem wird hier das aktuell verbundene Board mit dem dazugehörigen COM-Port angezeigt.
3. Icon-Leiste für die verschiedenen Anzeigemöglichkeiten im linken Panel: Explorer, Suche, Git, Debugging, Boards Manager und Library Manager.

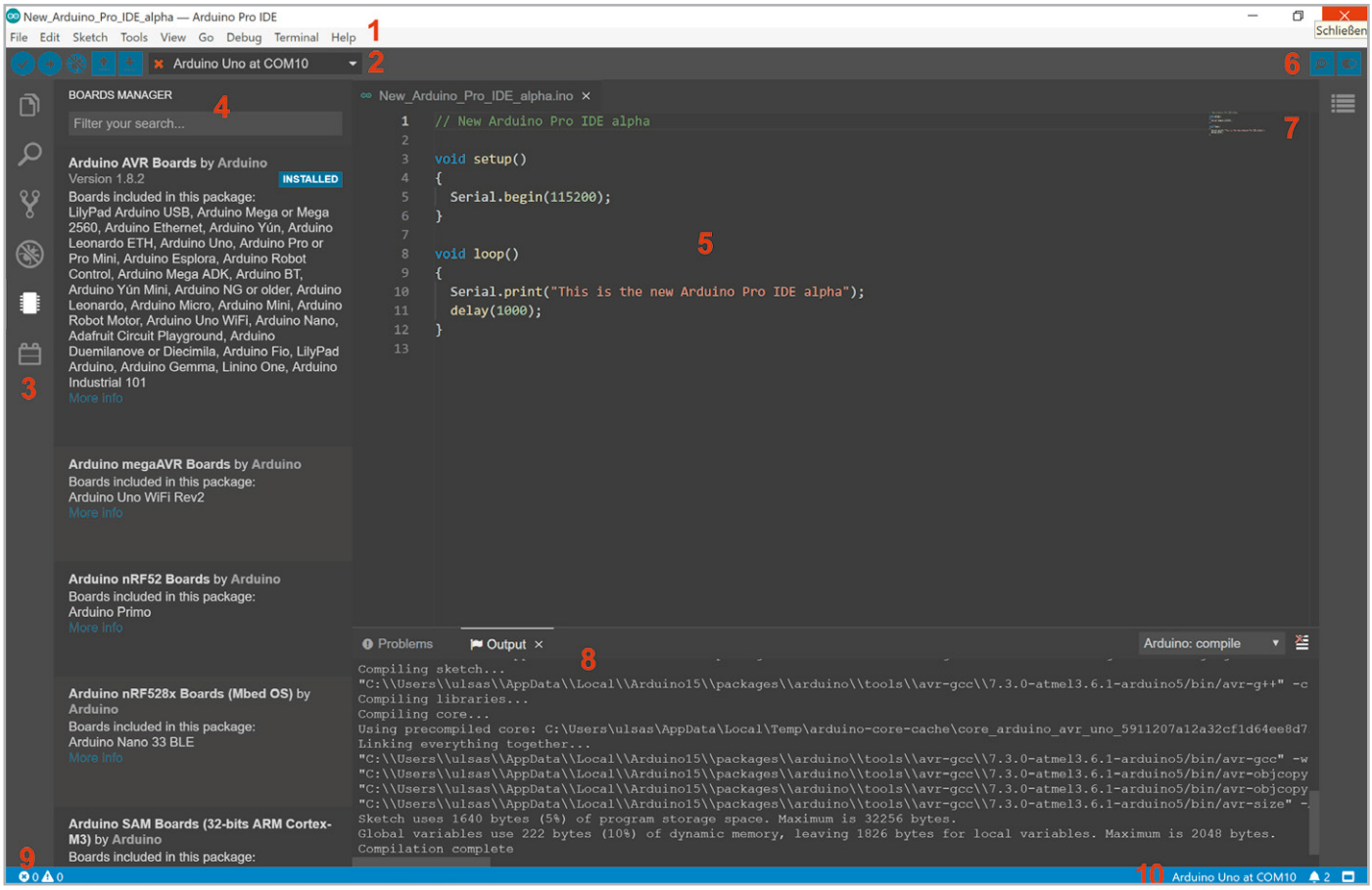


Bild 4: Die Oberfläche der Arduino Pro IDE im Advanced Mode

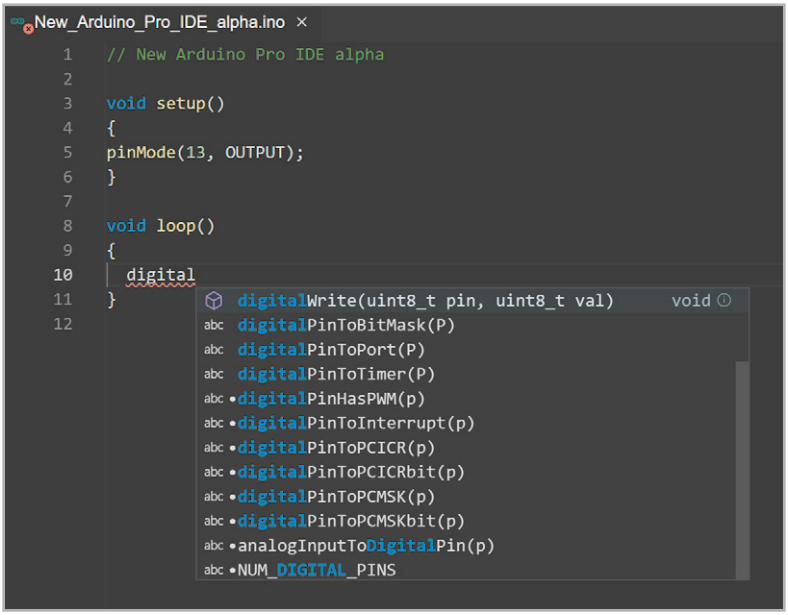


Bild 5: Code-Autovervollständigung

4. In diesem Panel werden die Details zu den jeweils o. g. ausgewählten Funktionen angezeigt.
5. In diesem Fenster wird der Code eingegeben.
6. Hier kann der serielle Monitor ein- und ausgeschaltet und vom normalen in den Advanced Mode umgeschaltet werden.
7. Ein kleines Fenster zeigt den gesamten Code in verkleinerter Form an (Minimap). Das hilft vor allem bei Fehlern im Code, deren Auftreten auf diese Weise schneller lokalisiert werden kann.
8. Im unteren Panel werden beispielsweise Probleme, Ausgaben des Compilers, der serielle Monitor und das Terminal dargestellt.
9. Hier wird angezeigt, in welcher Zeile man sich im Programm befindet und ob es Probleme/Fehler gibt.
10. Anzeige des angeschlossenen Entwicklungsboards, Benachrichtigungen und die Möglichkeit, das untere Panel ein- und auszuschalten.

Bei der Oberfläche wird schnell klar, dass hier eine professionelle Programmierumgebung entstanden ist. Die Einstellmöglichkeiten und Funktionen sind deutlich vielfältiger, die unter der Oberfläche steckenden Konfigurationen lassen zusätzliche Möglichkeiten der Individualisierung zu.

Kleine Hilfe – große Wirkung

Von professionellen Programmierumgebungen ist man gewohnt, dass ein eingegebener Code automatisch vervollständigt wird. Das funktioniert auch schon in der Alpha-Version der Arduino Pro IDE. Das ist zum einen praktisch, weil man Zeit bei der Eingabe spart, vor allem aber weil man überhaupt erst einmal sieht, welche verschiedenen Funktionen vorhanden sind. Außerdem werden die möglichen Argumente und deren Datentyp der Funktion aufgelistet (Bild 5).

In unserem konkreten Beispiel geben wir digital ein und bekommen eine Auswahl der vorhandenen Funktionen. Ausgewählt ist bei uns die Funktion digitalWrite(uint8_t pin, uint8_t val), die einen bestimmten Pin mit Spannung versorgt (HIGH) bzw. spannungslos schaltet (LOW). Ungewohnt dürfte den Arduino-Klassik-Nutzern die Bezeichnung des Datentyps uint8_t vorkommen, was im Grunde aber nichts anderes als ein Byte ist (Unsigned Integer mit einer Länge von 8 Bit = 1 Byte).

Verursacht man einen Fehler im Code, so schlägt die Arduino Pro IDE an verschiedenen Stellen in Echtzeit Alarm und dürfte damit vor allem Einsteigern eine große Hilfe gegenüber der klassischen IDE sein.

Unser Beispiel (Bild 6) zeigt ein vergessenes Semikolon (roter Kreis) – ein typischer Fehler nicht nur bei Anfängern. Gleich an fünf (!) Stellen warnt die IDE vor einem Fehler noch während der Eingabe und nicht wie bisher beim Kompilieren. Zudem gibt es im Panel

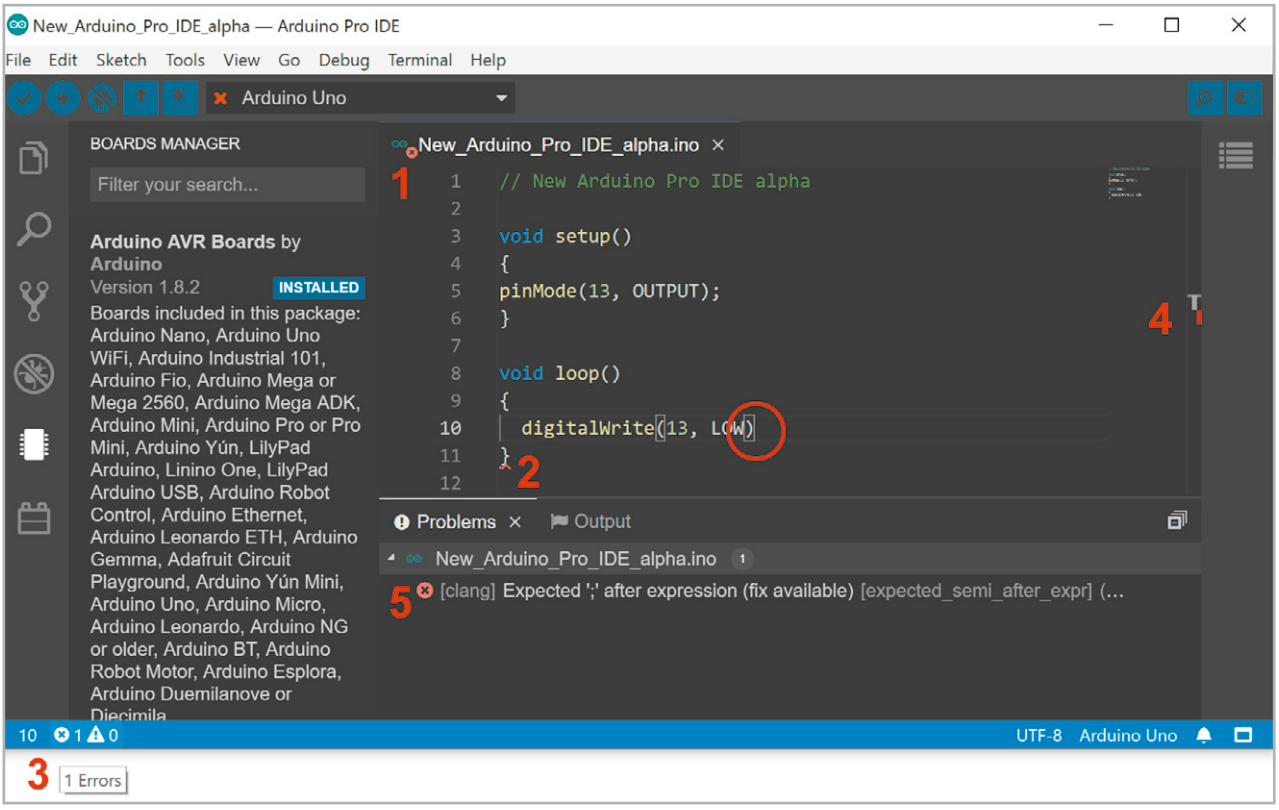


Bild 6: Zahlreiche Fehlerhinweise

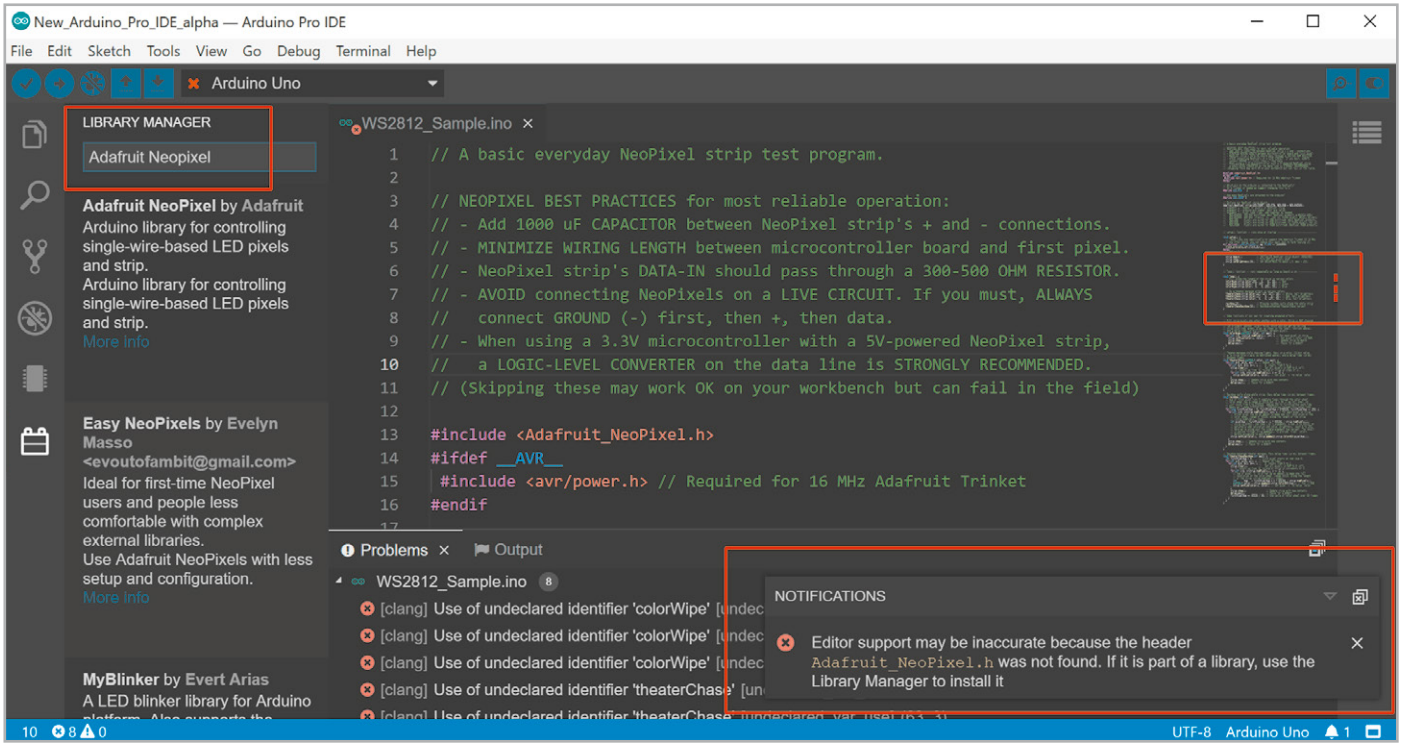


Bild 7: Hilfreiche Anzeige fehlender Bibliotheken

Problems einen Lösungsvorschlag. Geht man mit der Maus im Codefenster auf die rot geschlängelte Linie (in Bild 5 bei der geschweiften Klammer in der Loop-Funktion) gibt es per Pop-up ebenfalls einen Hinweis zur Problemlösung. Das ist vor allem praktisch, wenn man im unteren Panel gerade eine andere Funktion aufgerufen hat.

Bibliotheken, Boards, serieller Monitor

Eine weitere Verbesserung ist die direkte Anzeige des Bibliotheken-Managers. Hier kann man per Suchfunktion die verfügbaren Bibliotheken auflisten lassen und gleich installieren (linkes Fenster von Bild 7). Das ist vor allem praktisch, wenn man ein Projekt übernimmt, das viele Bibliotheken enthält. Zudem wird bei Aufruf eines Sketches ein Hinweis in Echtzeit angezeigt, welche Bibliothek fehlen könnte.

Im rechten Fenster von Bild 7 wird hier übrigens die hilfreiche Funktion der Minimap sichtbar. Ist der Code länger als das Code-Panel, gibt es eine Übersicht, an welchen Stellen im Code sich überall Fehler befinden.

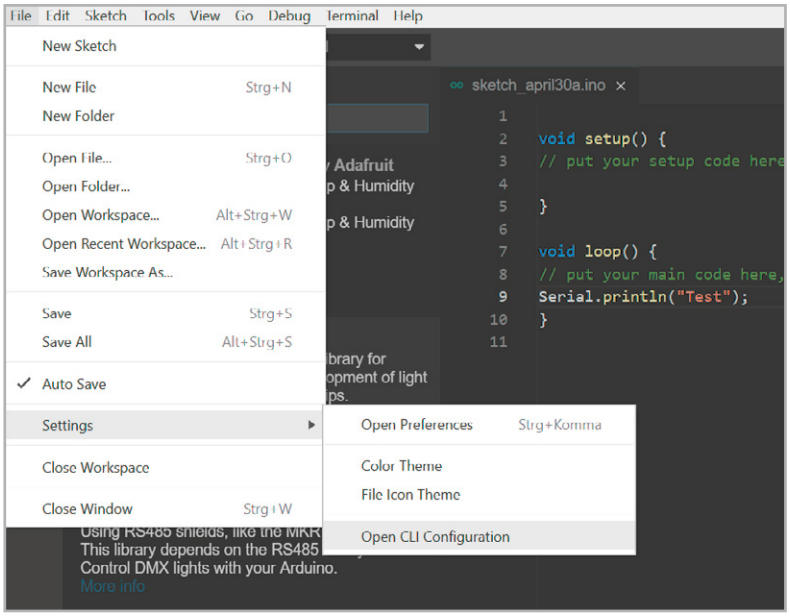


Bild 8: Über das Menü muss man die CLI-Konfiguration aufrufen …

Zurzeit lassen sich noch keine Archiv-Dateien (.zip) als Bibliotheken einbinden – das wird aber sicherlich schon bald hinzugefügt werden, da es sich dabei um eine wesentliche Funktion in der Programmierumgebung handelt. Übrigens gilt auch hier: Wurde eine neue Bibliothek installiert, muss die Programmierumgebung neu gestartet werden.

Auch die Möglichkeit zur Einbindung verschiedener Bibliotheksversionen, die man je nach verwendetem Sketch auswählen bzw. anpassen kann, konnten wir bisher nicht finden.

Beliebte Entwicklungsboards wie die, die auf Espressif SoCs ESP8266 oder ESP32 basieren, können ebenfalls eingebunden werden. Dazu geht man im Menü auf Settings -> Open CLI Configuration (Bild 8) und gibt dann unter additional_urls die entsprechende URL für das Drittanbieter-Board ein. Für die Unterstützung der ESP32-Boards also beispielsweise https://dl.espressif.com/dl/package_esp32_index.json (Bild 9).

Anschließend kann man die entsprechenden Boards im Drop-down-Menü Select Other Board & Port auswählen und gegebenenfalls im Menü Tools weitere boardspezifische Einstellungen vornehmen.

Der serielle Monitor lässt sich jetzt ebenfalls in der IDE direkt anzeigen, und zwar im unteren Fenster, das damit verschiedene Panels bzw. Registerkarten erhält, die man jeweils auswählen kann. Was bisher allerdings fehlt, ist die Möglichkeit, serielle Daten in einem Graph anzeigen zu lassen, wie es bei der klassischen Arduino IDE mit dem seriellen Plotter möglich ist.

Versionsverwaltung und Debugging

Will man als Team an einem Projekt arbeiten oder generell eine professionelle Versionsverwaltung verwenden, dürfte die Git-Integration gefallen. Hier kann man mit von Git unterstützen Systemen wie Git-

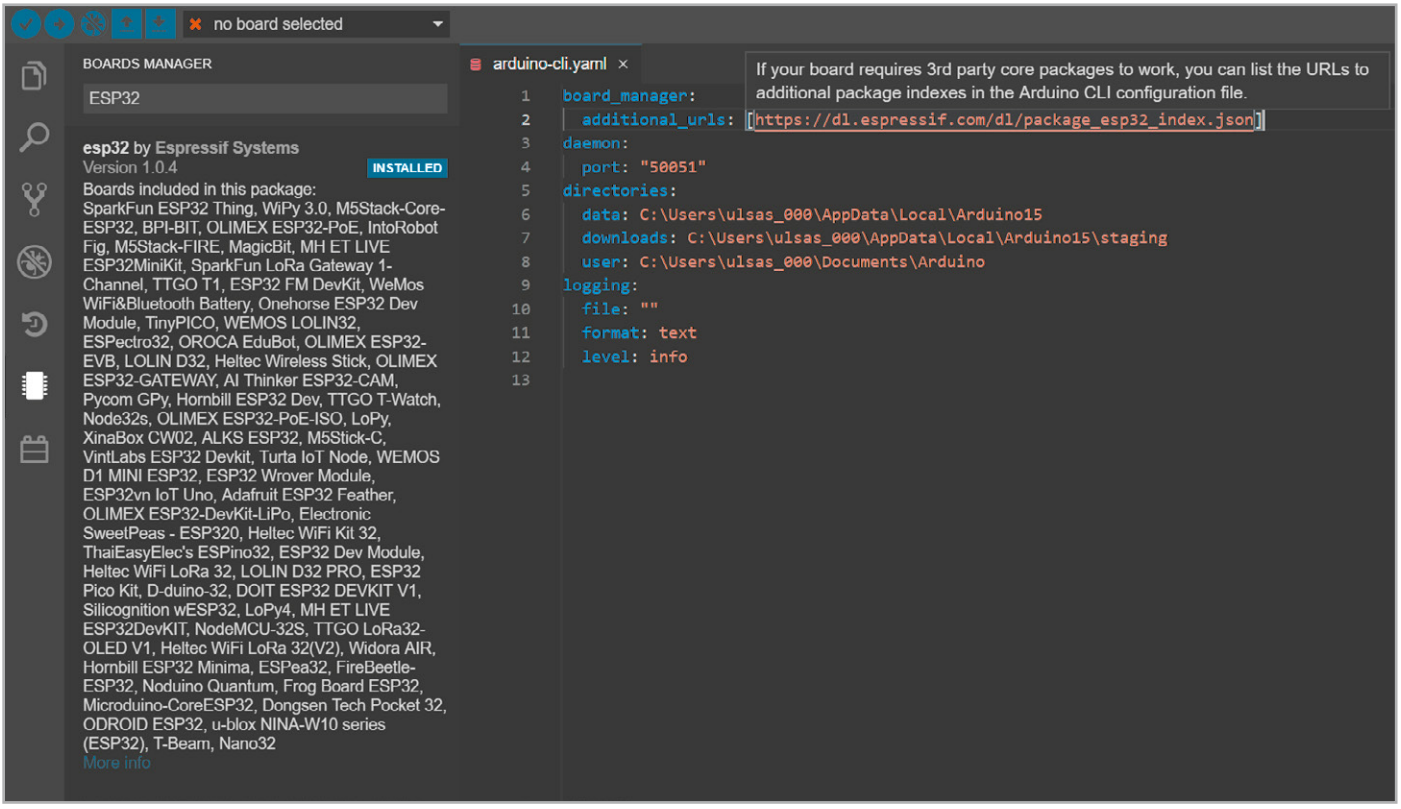


Bild 9: … und kann im nachfolgenden Fenster die URL für das Drittanbieter-Board eingeben.

Hub oder GitLab gemeinsam an Projekten arbeiten. In unserer Alpha-Version und angesichts der bisher nur sehr spärlichen Dokumentation werden wir in einem späteren Beitrag auf die Möglichkeiten der Versionsverwaltung näher eingehen und diese testen.

Gleiches gilt für das Debugging. Momentan werden nur Entwicklungsboards wie das Arduino Zero [9] mit zusätzlichem Debugging IC unterstützt. Generell soll das Debugging zukünftig nur mit ARM-Boards funktionieren. Damit bleiben beispielsweise die klassischen Boards wie UNO, Nano, Micro, die auf dem ATmega328p bzw. ATmega32U4 basieren, leider außen vor.

Beim Debugging soll das Setzen von Breakpoints und die schrittweise Abarbeitung des Codes möglich sein. So kann man zeilengenau sehen, wie der Code abgearbeitet wird, sich entsprechende Variablen ausgeben lassen und vor allem das bisherige Debugging per Serial.print(), das unter Umständen das Timing im Code und damit die Ausgabe beeinflusst hat, vermeiden.

Sobald hier weitere Boards nutzbar sind, werden wir uns das Thema Debugging mit der Arduino Pro IDE genauer anschauen.

Fazit

Für eine Alpha-Version, die es erst seit Oktober 2019 offiziell gibt, läuft die Programmierumgebung schon erstaunlich stabil. Mit dem Unterbau des Arduino CLI und dem Frontend aus der mit Webtechnologien arbeitenden Kombination Theia und Electron hat man ein universelles Programmiertool geschaffen, welches nach 15 Jahren Arduino IDE die Bezeichnung Programmierumgebung auch für den semi-professionellen oder Profi-Bereich verdient.

Die Features, die man ausgewählt hat, ergeben alle Sinn und sind zum Teil schon in der vorliegenden Version sinnvoll nutzbar. Auto-Vervollständigung, Echtzeit-Fehleranalyse, Debugging und Git-Integration werden noch mehr (semi-)professionelle Anwender in das Arduino-Ökosystem ziehen. Und sei es nur für die Prototyping-Phase. Die kürzlich realisierte Möglichkeit zur Einbindung von Drittanbieter-Boards eröffnet zudem die IDE für viele weitere Nutzer.

Dass das Debugging noch nicht für weitere Boards möglich ist, ist zwar ein Wermutstropfen, der mit Erscheinen dieses Beitrags aber schon Geschichte sein könnte.

Die große Community, die hinter Arduino steckt, wird ihr Übriges tun, um die Entwicklung der Arduino Pro IDE weiter voranzutreiben. Dass der Open-Source-Gedanke nicht verloren geht und durch Arduino CLI einen in vielen weiteren Tools nutzbaren Unterbau bekommt, der zudem plattform- und rechnerübergreifend genutzt werden kann, kann man zusätzlich zur vorliegenden Version im Alpha-Stadium nur positiv anmerken. ELV

Weitere Infos:

[1] Website Arduino: www.arduino.cc
[2] Git: git-scm.com
[3] Atmel Studio (Microchip): www.microchip.com/mplab/avr-support/atmel-studio-7
[4] PlatformIO IDE: platformio.org/
[5] Arduino Pro IDE auf GitHub: github.com/arduino/arduino-pro-ide
[6] Blogbeitrag Arduino Command Line Interface (CLI): blog.arduino.cc/2020/03/13/arduino-cli-an-introduction/
[7] Theia Framework: theia-ide.org/
[8] Electron: www.electronjs.org
[9] Arduino Zero: store.arduino.cc/arduino-zero

Alle Links finden Sie auch online unter de.elv.com/elvjournal-links