Einrichten der Arduino IDE und Einbindung der ESPs sowie die Einrichtung von Thonny unter macOS

An Apple a day keeps Windows away :-)

Teil 1 – Arduino einrichten

Rudolf Reiber
11. März 2021

1. Eine kurze Vorbemerkung

Auch Benutzer von Apples macOS Rechnern verspüren hin und wieder das Bedürfnis, sich mit Mikrocontrollern zu beschäftigen. Dazu gibt es viel Literatur, auch findet man im Netz eine große Anzahl von Beiträgen. Doch bezieht sich die überwiegende Anzahl dieser Informationen auf die Nutzung der Controller unter Windows Rechnern. Die Maclaner stehen wohl im Ruf, nichts selber machen zu wollen oder zu können und sich nur mit Grafik zu beschäftigen. Das mag zwar im Mittel zutreffen, stimmt aber nicht für alle »Maccerer«. Dies und natürlich auch die relativ geringe Verbreitung der Macs ist vermutlich der Grund für die eher stiefmütterliche Behandlung dieser Plattform bei diesem Thema.
Dieser Beitrag soll helfen, Einsteigern bei der Einarbeitung in dieses schöne Gebiet etwas zur Hand zu gehen.

2. Hilfreiche Software

Dieser Beitrag wurde unter macOS »Big Sur« (Version 11.2.2) geschrieben, dem aktuell neuesten Betriebssystem.
Zunächst möchte ich einen kurzen Überblick über hilfreiche Software geben. Die Aufzählung ist natürlich nicht vollständig, sondern den Vorlieben des Autors geschuldet.

Systemsoftware

a) »**Terminal**« zum Aufruf diverser Shells um per Konsolenbefehle mit dem Rechner zu interagieren.
b) »**Vorschau**«, der oft unterschätzte PDF-Reader von macOS mit einfachen Möglichkeiten zur Bearbeitung von PDFs und Bildern, reicht für den Hausgebrauch.
c) »**Safari**«, der mitgelieferte Browser
d) »**Textedit**«, ein einfacher Texteditor, mitgeliefert

Software von »außerhalb«

a) »**iTerm2**«, eine sehr mächtige Terminalsoftware, kostenlos erhältlich unter: https://iterm2.com
b) »**BBEdit**« Sehr mächtiger Texteditor, kostenpflichtig ($50), auch im AppStore, nur in Englisch erhältlich
c) »**CotEditor**«, einfacher Texteditor, kostenlos im AppStore
d) »**Firefox**«, »**Google Chrome**«, »**Opera**«, »**Vivaldi**«, »**Microsoft Edge**«, allesamt Browser, je nach persönlichem Geschmack wählbar

3. Installation der Arduino IDE

Voraussetzung zur Programmierung der Arduino oder ESP-Module ist mindestens eine Entwicklungsumgebung. Mir sind davon zwei bekannt, zum Einen die IDE von Arduino selbst und zum Anderen das »Visual Code Studio« von Microsoft. Für den Anfänger halte ich die »Arduino IDE« bestens geeignet, da zwar nicht so mächtig, aber weniger kompliziert zur Einarbeitung.
Die Arduino IDE gibt es kostenlos auf der Website von Arduino:
https://www.arduino.cc/en/software
Hier gibt die Versionen für Windows, LINUX und macOS, zur Zeit der Erstellung dieses Textes die Version 1.8.13. Nach dem Klick auf die Zeile »Mac OS X« und der Beantwortung der Frage ob wir nur laden oder vorher eine kleine Spende geben wollen, wird die Software geladen und landet im Ordner »Downloads«, wo sie gleich entpackt wird.

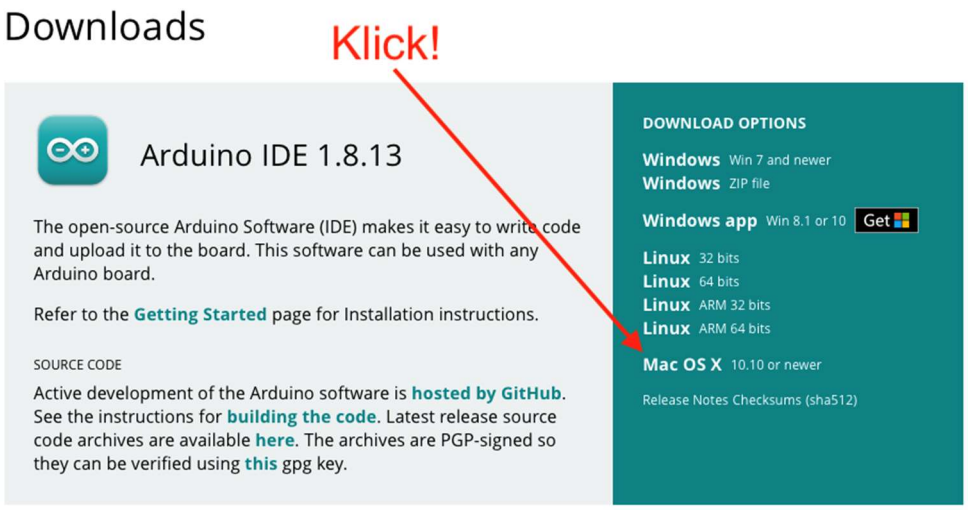

Bild 1

Von dort ziehen wir das Programm in den »Programme«-Ordner.

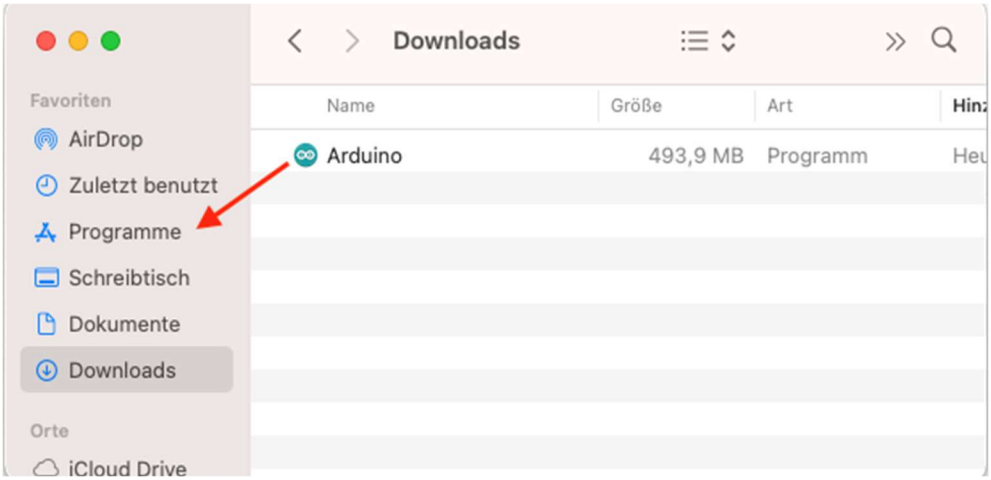

Bild 2

Nun wird das Programm »Arduino« gestartet. Zunächst müssen wir die einmalige Sicherheitsabfrage beantworten, da das Programm nicht aus dem AppStore kommt. Immerhin scheinen die Entwickler bei Apple registriert zu sein.

Bild 3

Jetzt kommt das »Arduino«-Fenster:

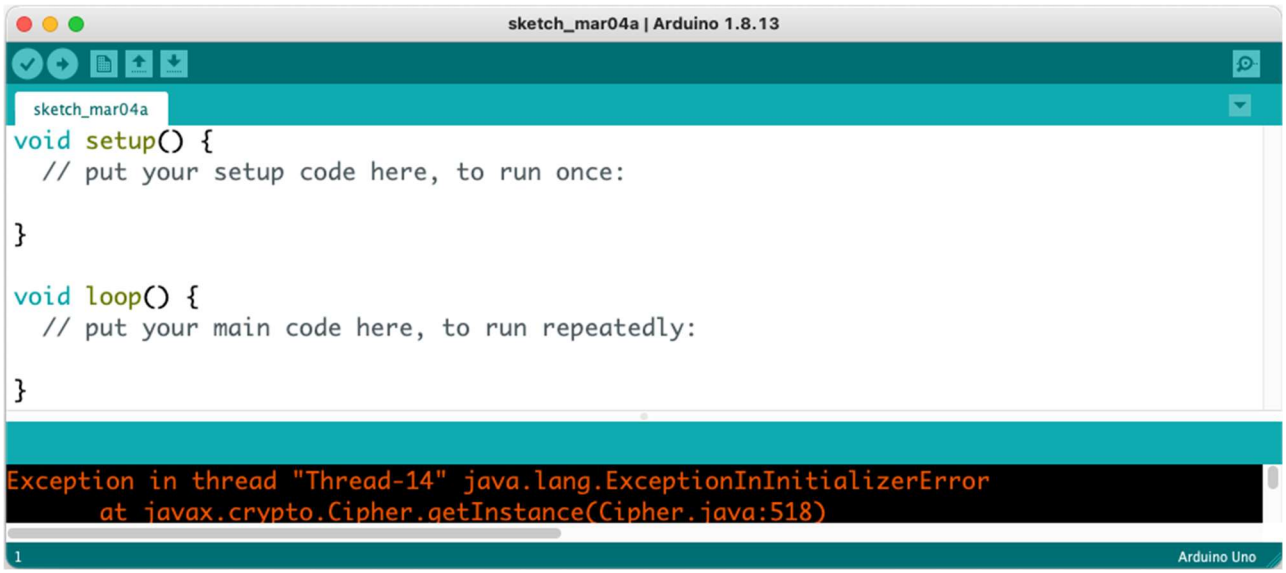

Bild 4

Die roten Fehlermeldungen im Fenster unten können ignoriert werden, da noch kein-Board angeschlossen ist.
Eine Installation von Treibern zum Anschluss der Boards war bei mir nicht nötig.
Folgende Boards:

- »Arduino MEGA® 2560 REV3« von Arduino
- »Mega 2560 R3 Board mit ATmega2560« von AZ-Delivery
- »ATmega328 ENTWICKLUNGSBOARD« von Vellemann
- »Mikrocontroller Board mit ATmega328P, ATmega16U2, kompatibel mit Arduino UNO® R3« von AZ-Delivery
- »Nano V3.0 mit Atmega328 CH340« von AZ-Delivery

funktionierten ohne weitere Treiberinstallation.
Beim ersten Start wird im Benutzerordner ein Ordner »Arduino« angelegt. In diesen Ordner werden dann alle Projekte abgelegt.

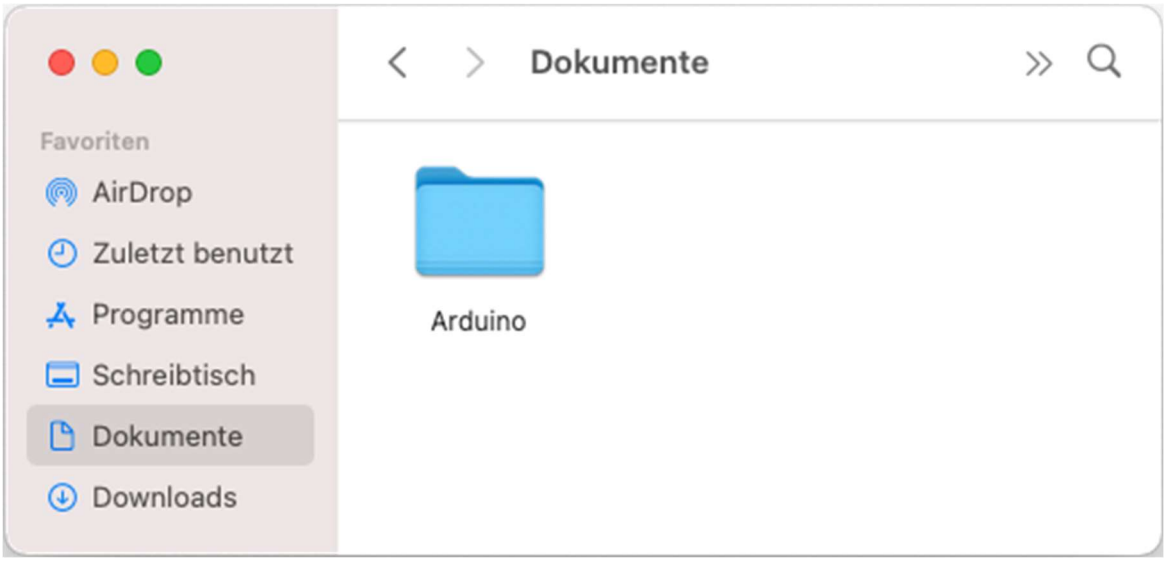

Bild 5

Wenn weitere Bibliotheken geladen werden, kommen diese automatisch in einen Ordner »libraries«, der im Ordner »Arduino« liegt.
Der UNIX-Pfad zu diesen Ordnern ist:
~/Documents/Arduino bzw. ~/Documents/Arduino/libraries
»~« ist die Abkürzung für »/Users/benutzer«. (»benutzer« ist natürlich die Bezeichnung des jeweiligen Nutzers. (Das Zeichen »~« erhält man mit »option-n«.)

4. Einrichten der IDE

IDE bedeutet »Integrierte Entwicklungsumgebung« vom englischen »integrated development environment« und ist eine Sammlung von Programmen, mit der eine einfache Softwareentwicklung ermöglicht werden soll.
Die Arduino-IDE kann in gewissem Umfang angepasst werden. Sehen wir uns das an. Das Programm wird gestartet und der Menüpunkt »Arduino —> Preferences… « aufgerufen.

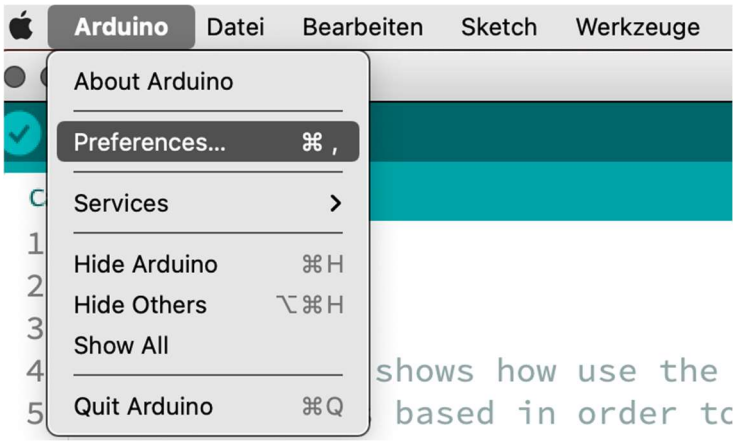

Bild 6

Das »Preferences«-Fenster:

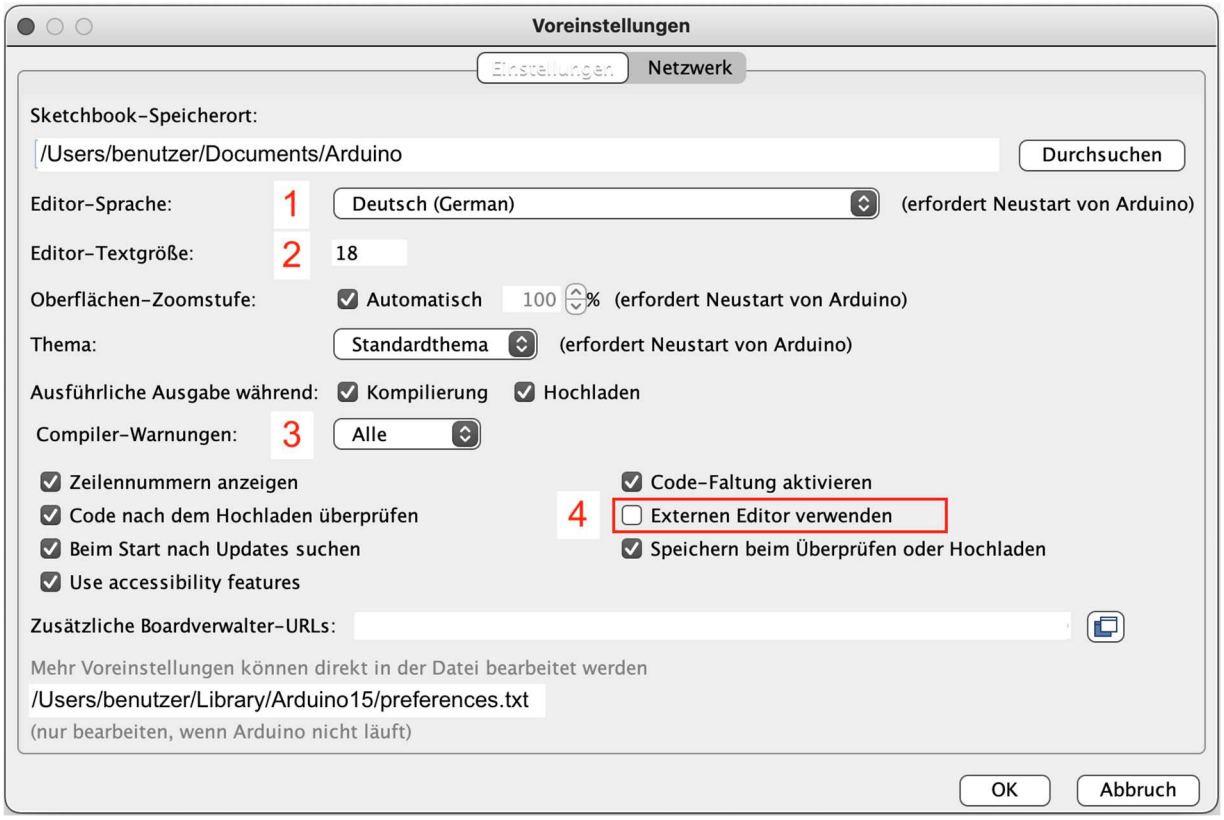

Bild 7

(1) Einstellen der Sprache
(2) Textgröße, für ältere Menschen empfehlen sich 18 Punkte. ;-)
(3) Einstellmöglickeiten: »Keine«, »Standard«, »Weitere« oder »Alle«. Für den Anfang »Standard« wählen.
(4) Wenn hier der Haken gesetzt ist, kann man die Programmtexte in einem externen Editor, wie z.B. CotEditor oder BBEdit schreiben. Das hat zum Einen Vorteile, weil diese Editoren viel mächtiger sind als der Arduino Editor, hat aber zum Anderen den Nachteil, dass man zwei Fenster geöffnet hat und dass die Syntaxhervorhebung, wie sie die Arduino-IDE macht, nicht zur Verfügung steht. Ausprobieren und entscheiden, was einem besser gefällt.

5. Das erste Programm

Für die ersten Schritte verwende ich das Mikrocontroller Board mit ATmega328P, ATmega16U2, kompatibel mit Arduino UNO R3, das mit einem USB-Kabel an den Mac angeschlossen wird. Dann geht es los:

a) Arduino Programm starten
b) Die IDE liefert eine Vielzahl von Beispielprogrammen mit, die hier »Sketche« genannt werden. Eines davon, »Blink«, werden wir zum Leben erwecken.
Nomen est Omen, das Programm lässt die eingebaute LED an Pin 13 im Sekundentakt aufleuchten.
c) Wir öffnen den Sketch »Blink«

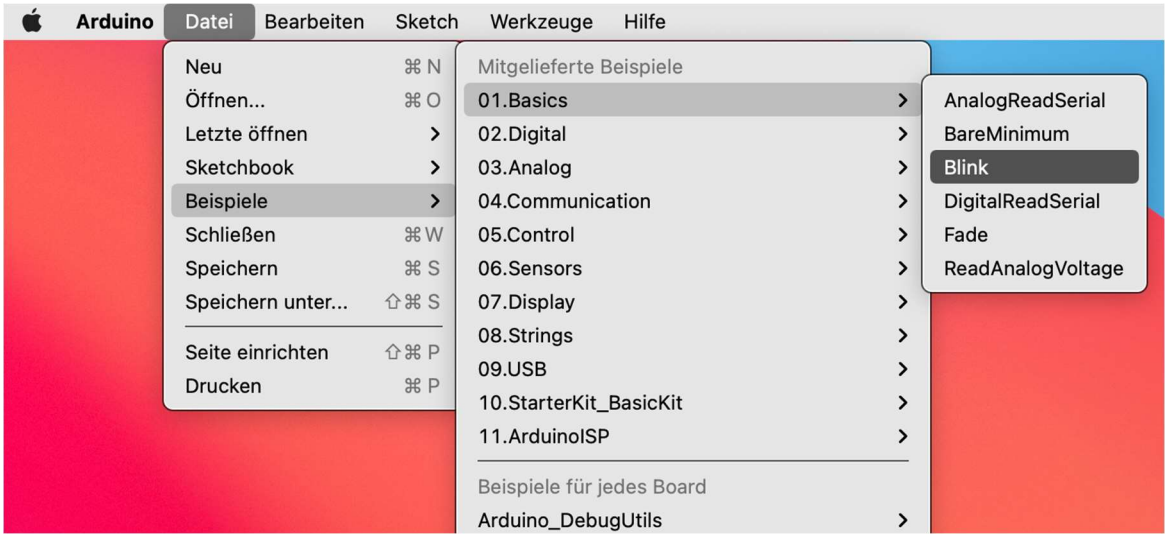

Bild 8

Der Sketch »Blink«

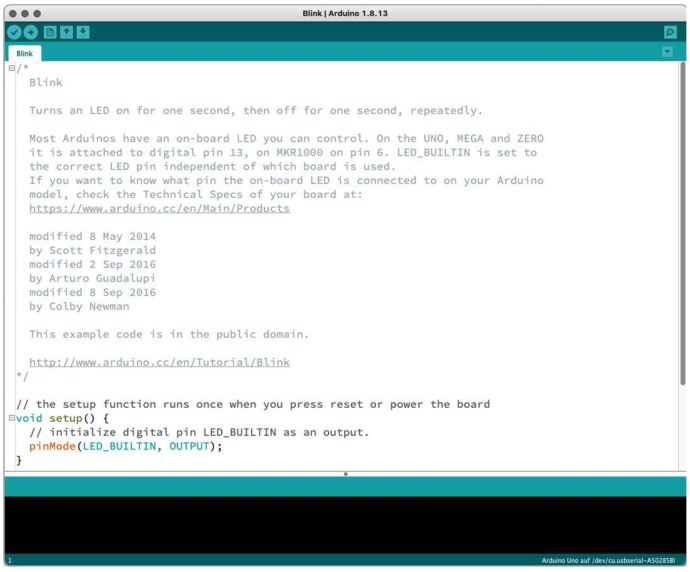

Bild 9

d) Und wählen den Typ des angeschlossenen Boards aus

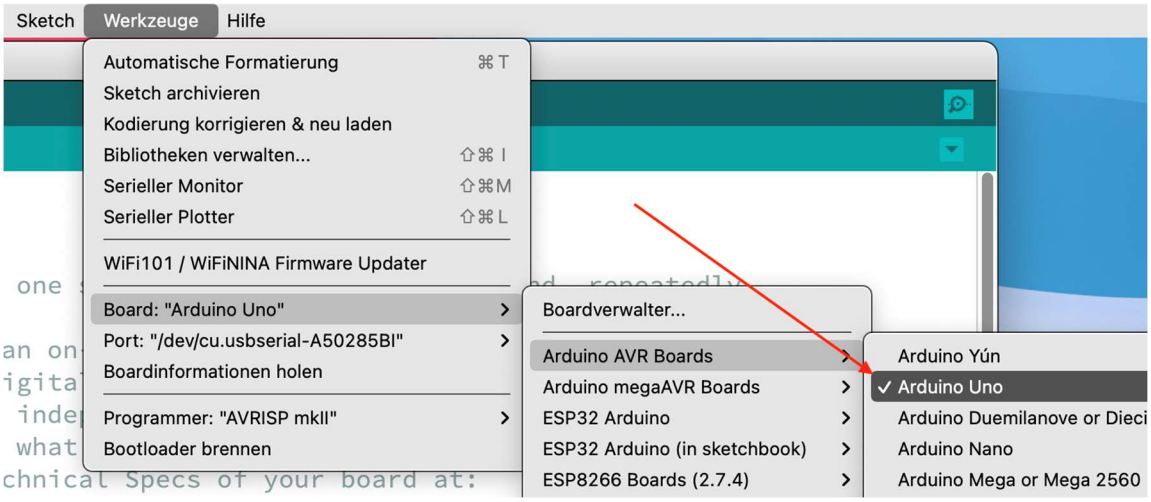

Bild 10

e) Einstellen der Ports
Die Bezeichnung für den Port hängt vom verwendeten USB-Controllerchip des Boards ab, beginnt aber immer mit »/dev/cu.usbserial-«.

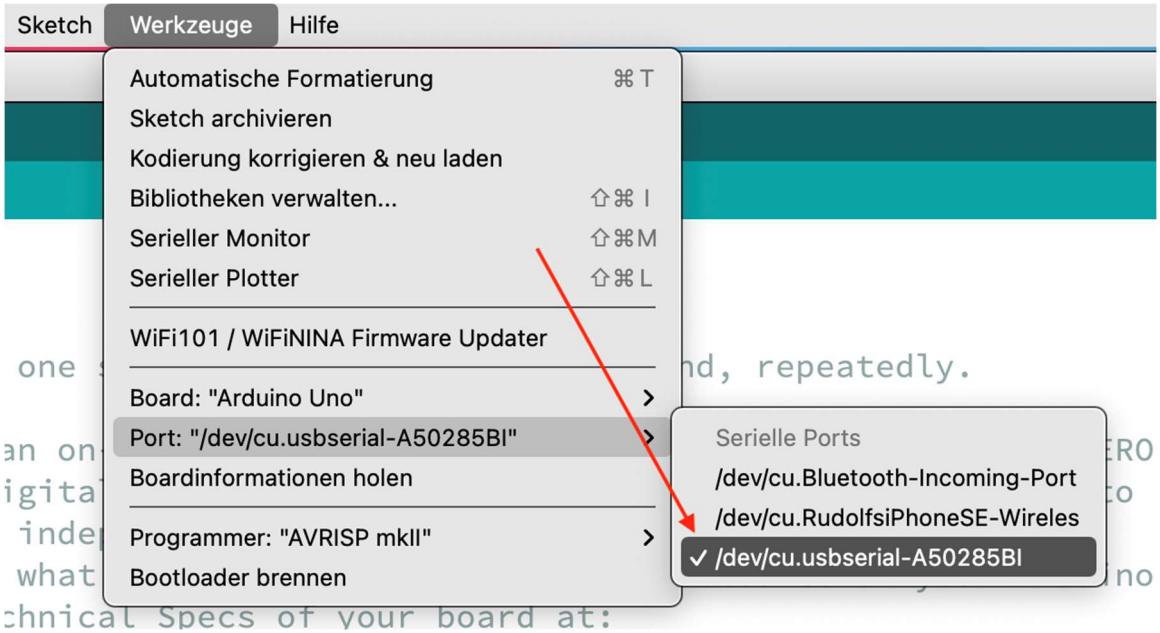

Bild 11

f) Jetzt noch übersetzen und auf das Board spielen
Klickt man auf den Knopf mit dem Haken, wird das Programm nur übersetzt. So kann man feststellen, ob ein Syntaxfehler vorliegt, logische Fehler werden so natürlich nicht gefunden. :-(
Ein Klick auf den Knopf mit dem Rechtspfeil übersetzt das Programm in eine Version, die

der Mikrocontroller ausführen kann und überträgt den Programmcode auf den Chip des Boards.

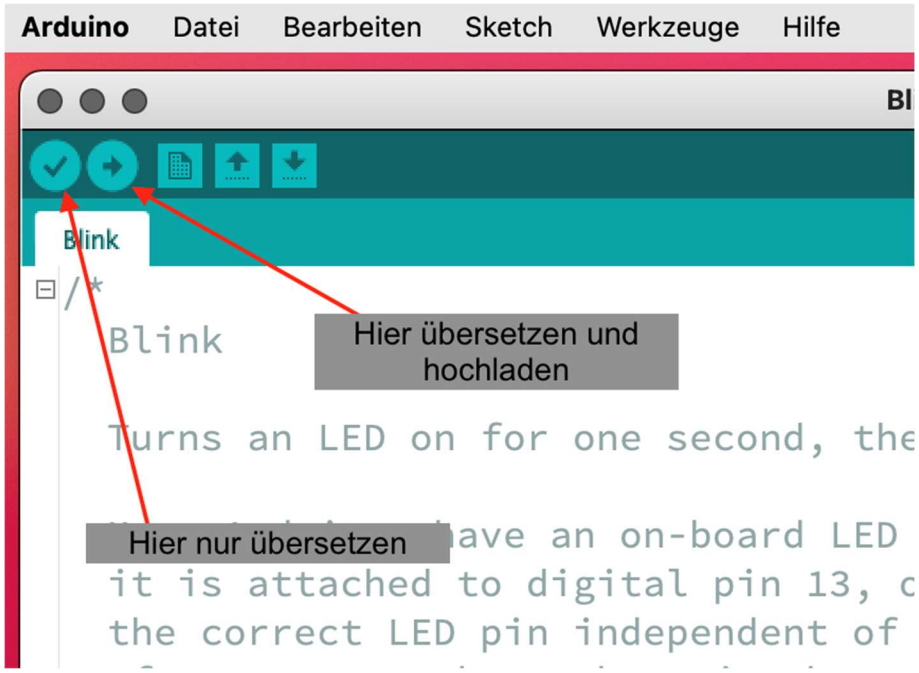

Bild 12

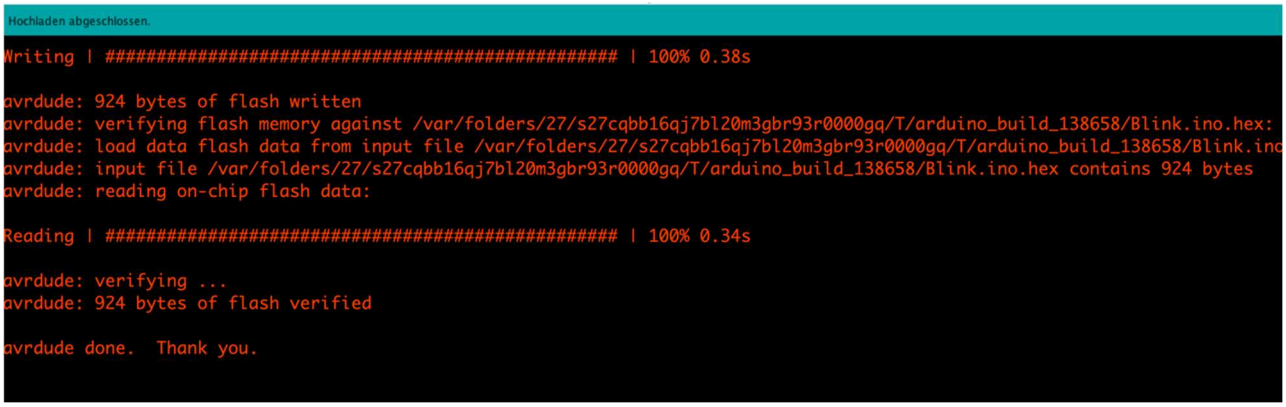

Bild 13

In roter Schrift steht die Erfolgsmeldung unten im Fenster.

g) Nach kurzer Zeit sollte die LED fröhlich blinken.

Geschafft!

h) Ein Hinweis zu den mit dem Arduino NANO kompatiblen Boards:
Es gibt Boards, die mit noch dem »alten Bootloader« versehen sind, andere haben den neuen Bootloader. Das Board »Nano V3.0 mit Atmega328 CH340« von AZ-Delivery wird mit dem alten Bootloader ausgeliefert. Deshalb muss in der Arduino IDE eine weitere Einstellung vorgenommen werden, denn der passende Bootloader muss extra ausgewählt werden:

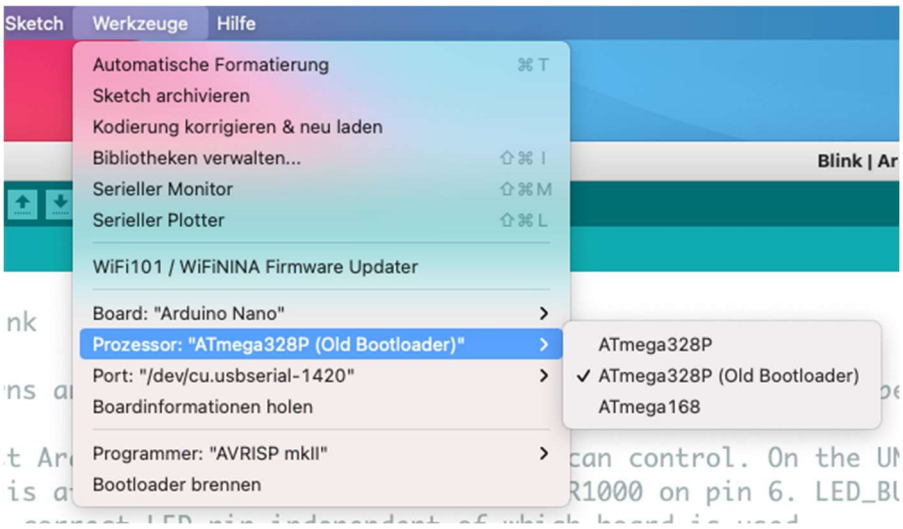

Bild 14

Wer den neuen Bootloader auf des Board spielen will, findet hier eine gute Anleitung dazu:
https://www.az-delivery.de/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/az-delivery-nano-v3-bootloader-flashen
Bei mir war der Kondensator von 10µF zwischen GND und RST nötig, damit der Flashvorgang klappte.

Beim nächsten Mal integrieren wir die Micro Controller von Espressif in die Arduino IDE.