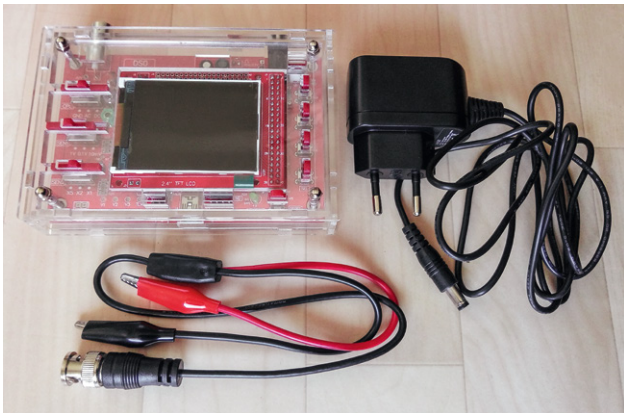
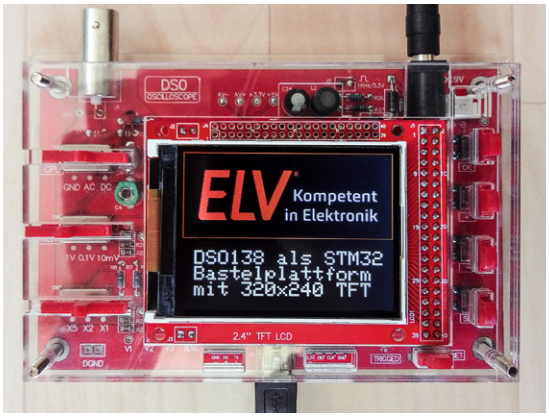
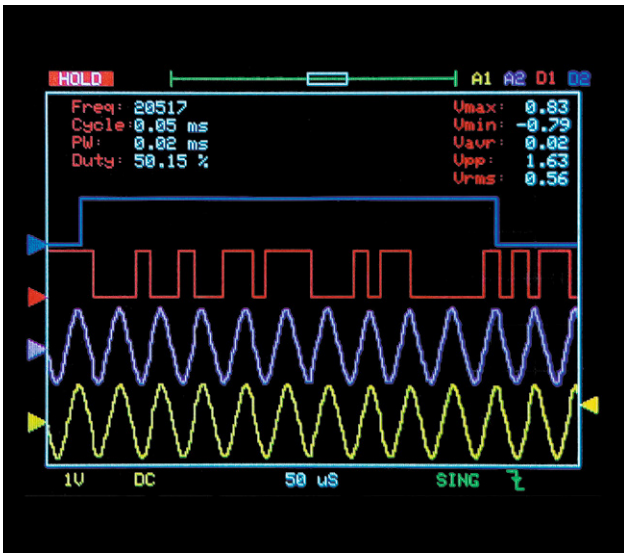
Besser machen

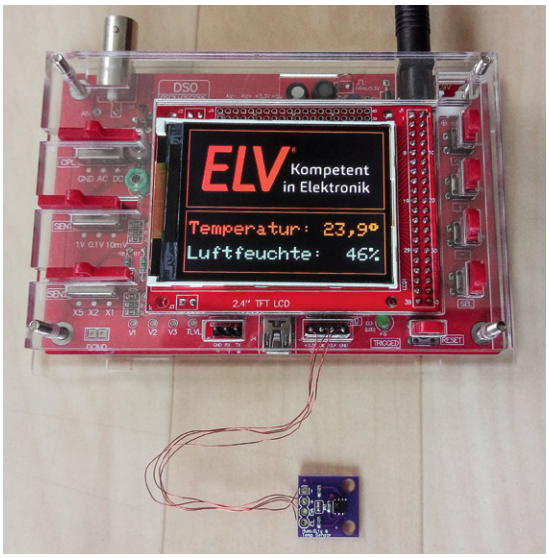
STM32 mit Arduino IDE nutzen und Mini-Oszilloskop DSO138 optimieren, Teil 1

Mit der Arduino IDE steht eine Programmierplattform für die Arduino-Entwicklungsboards, die hauptsächlich auf Microchip-Prozessoren (ehemals Atmel-Prozessoren) basieren, zur Verfügung. Mit der Zeit sind auch andere Mikrocontroller wie ESP8266/ESP32 von Espressif in diese Entwicklungsumgebung integriert worden. Wir zeigen in unserem praktischen Beispiel anhand der Programmierung des kostengünstig erhältlichen Einsteiger-Oszilloskops DSO138, wie man auch mit STM32-/STM8-Controllern von STMicroelectronics diese Plattform nutzen kann. Dabei lernen wir nicht nur mehr über die notwendige Toolchain, sondern verbessern auch noch die Funktionen des DSO138 mit einer Open-Source-Firmware.









Hinweis: *Alle hier vorgestellten Modifikationen geschehen auf eigenes Risiko und eigene Gefahr. Außerdem erlöschen sämtliche Garantieansprüche.*

Mini-Oszilloskop DSO138

Viele Hobby-Elektroniker könnten hin und wieder ein Oszilloskop gebrauchen, um Signalverläufe grafisch anzeigen zu können. Um das Budget zu schonen, bietet sich ein günstiges Mini-Oszilloskop an, z. B. das DSO138 [1], das für viele Anwendungsfälle ausreicht. Es bietet eine Samplerate von 1 MS/s bei einer Analogbandbreite bis 200 kHz. Die Messauflösung beträgt 12 Bit, und es können 1024 Messwerte aufgezeichnet werden.

Sehr interessant an diesem Modell ist die Tatsache, dass der Schaltplan frei verfügbar ist [2]. Dort sieht man, dass ein Mikrocontroller des Typs STM32F103 als Steuer- und Messzentrale verbaut ist. Die Firmware ist im Onchip-Flash des STM32 gespeichert. Dieser Programmspeicher kann über einen herstellerseitig eingebauten Bootloader, der sich in einem nicht veränderbaren ROM-Bereich des Controllers befindet, beschrieben und gelesen werden. Hierzu ist ein USB-UART-Modul erforderlich.

Bei dem bei ELV erhältlichen DSO138-Modell von JOY-iT ist der Ausleseschutz des Flashs konfiguriert, sodass sich die im Auslieferzustand vorhandene Oszilloskop-Firmware leider nicht sichern lässt. Auf dem Startbildschirm des DSO138 wird jedoch die vorhandene Firmware-Version angezeigt, und da einige Versionsstände auf der Hersteller-Website (s. u.) bereitgestellt werden, kann auch ohne vorherige Sicherung später diese Version wieder aufgespielt werden.

Da für die STM32-Familie die Arduino-Erweiterung STM32duino zur Verfügung steht, kann man zudem recht einfach eigene Firmwareprojekte entwickeln. Für das DSO138 wurde damit sogar eine alternative Firmware namens DLO-138 als Open-Source-Projekt entwickelt, die noch mehr Features bietet als die Original-Firmware. Außerdem lässt sich dank STM32duino das DSO138 auch als preisgünstige Experimentierplattform mit TFT einsetzen, wobei auch jederzeit wieder die Oszilloskop-Firmware aufgespielt werden kann.

Arduino mit STM32 – STM32duino

Der auf dem DSO138 verbaute STM32F103C8 ist ein leistungsfähiger 32-Bit-Mikrocontroller mit ARM-Cortex-M3-Core, der mit bis zu 72 MHz getaktet werden kann. Über eine interne PLL wird der Takt des extern angeschlossenen Quarzes (hier 8 MHz) entsprechend hochgetaktet. Der Programmspeicher befindet sich im Onchip-Flash und hat bei der C8-Variante eine Größe von 64 kB. Dieser kann dank des eingebauten Bootloaders über den USART1 programmiert werden. Als Arbeitsspeicher stehen 20 kB RAM zur Verfügung. Außerdem bietet der Controller eine Vielzahl von Funktionen wie beispielsweise A/D-Wandler, USART, USB, SPI, I²C und Timer.

Die Programmierung eines solchen Mikrocontrollers kann komplex werden, da eine Vielzahl von Registern konfiguriert werden muss. Das „Reference Manual" der STM32F1-Familie umfasst mehr als 1000 Seiten [3].Es gibt jedoch eine Erweiterung für die Arduino IDE namens STM32duino, sodass auch ohne tiefgreifende Kenntnisse der STM32-Controller-Programmcode entwickelt werden kann.

Ursprünglich wurde das STM32duino-Projekt von Roger Clark als Weiterentwicklung des Codes für das Maple-Board von Leaflabs [4] begonnen, dessen Produktsupport 2016 vom Hersteller eingestellt wurde. Dabei wurden zunächst nur STM32F103-Boards unterstützt, also weder andere Varianten der STM32F1-Familie noch andere STM32-Familien wie z. B. STM32F4. Im Jahr 2018 übernahm dann STMicroelectronics das STM32duino-Projekt [5].

Der Programmcode wurde jedoch komplett umgebaut und als Aufsatz auf das Cube-HAL-Framework von STMicroelectronics konzipiert. Dadurch werden auch andere STM32-Familien unterstützt. Neben den Nucleo- und Discovery-Boards von ST werden auch viele andere STM32 Boards wie beispielsweise die bekannte „Blue Pill" unterstützt.

Parallel zum offiziellen STM32duino-Projekt von ST wird auch das ursprüngliche Projekt unter dem Namen Arduino_STM32 von Roger Clark weiterhin gepflegt [6].

Dieser Arduino-Core ist jedoch anders aufgebaut und basiert nicht auf dem CMSIS (Cortex Microcontroller Software Interface Standard) der die herstellerunabhängige Hardware-Abstraktionsschicht für Mikrocontroller darstellt, die auf Arm-Cortex-Prozessoren basieren.

Die Konstanten für die Register und Bitmasken heißen dort anders, d. h., ein vorhandener Arduino-Code für Arduino_STM32 ist oftmals nicht kompatibel mit dem offiziellen STM32duino-Core und erzeugt dort Compiler-Fehlermeldungen.

In den Header-Dateien auf [7] sind die unterschiedlichen Bezeichnungen ersichtlich. Beispielsweise wird bei Arduino_STM32 die Bezeichnung Timer2 verwendet, wohingegen bei STM32duino die Bezeichnung TIM2 gewählt wurde. Die später vorgestellte alternative Firmware DLO-138 für das DSO138 benötigt den Arduino_STM32-Core von Roger Clark. Es können problemlos beide STM32-Cores parallel installiert werden.

In den folgenden Schritten zeigen wir, wie man sich die Toolchain für Mikrocontroller auf Basis von STM32 unter der Arduino IDE einrichtet und darauf aufbauend eine neue Firmware auf das DSO138 aufspielt.

Schritt 1

Arduino als portable Version installieren

Im ersten Schritt wird die Arduino IDE als sogenannte Portable Version installiert. Dies hat den Vorteil, dass eine evtl. bereits bestehende

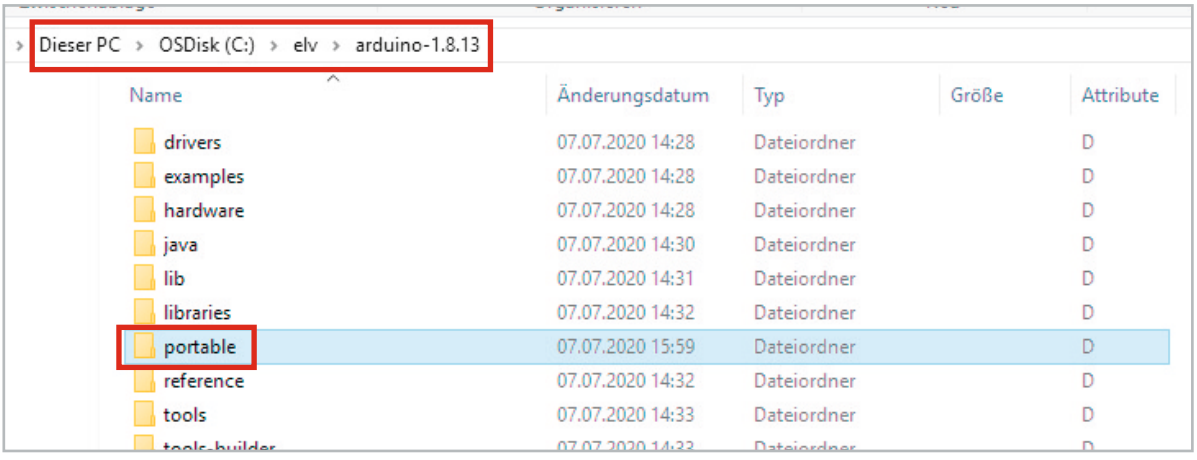


Bild 1: Manuelles Anlegen des Ordners portable

Arduino-Installation nicht angetastet wird und alle Erweiterungen unterhalb eines einzigen Ordners installiert werden anstatt verteilt in diversen Windows-Benutzerordnern. Zunächst wird auf [8] die aktuelle Version 1.8.13 als Zip-Datei heruntergeladen und in einen Ordner entpackt. Im Beispiel ist das C:\elv\arduino-1.8.13. Anschließend wird in diesem Ordner manuell ein Ordner namens „portable" angelegt (Bild 1).

Alle zukünftig über die Arduino IDE installierten Erweiterungen werden dann automatisch in diesem Ordner „portable" gespeichert, z. B. Sketchbook, Bibliotheken, Board-Packages und Hardware-Ordner. Die fertige Installation kann auch auf einen USB-Stick kopiert und auf einem anderen Rechner ausgeführt werden, ohne dort alles neu installieren zu müssen.

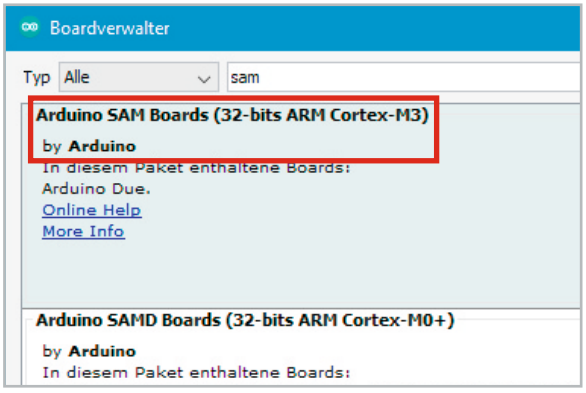


Bild 2: Installation des Pakets für Arduino-SAM-Boards

Schritt 2

ARM-Toolchain installieren

Nun wird im Ordner C:\elv\arduino-1.8.13 die Datei arduino.exe gestartet. Im Menü wird Werkzeuge → Board → Boardverwalter ausgewählt und dort das Board „Arduino SAM Boards (32-bits ARM Cortex-M3)" installiert (Bild 2).

Dieses Paket ist eigentlich für Arduino-Boards mit Atmel-SAM-Controllern, wie z. B. das DUE-Board gedacht. Es wird dadurch jedoch die Toolchain arm-none-eabi-g++ installiert, die dem Kompilieren von Codes für ARM-Cortex-Mikrocontroller dient und auch für den STM32 benötigt wird. Die Installation dauert einige Zeit. Am Ende erscheint der Hinweis „Installed" (Bild 3).

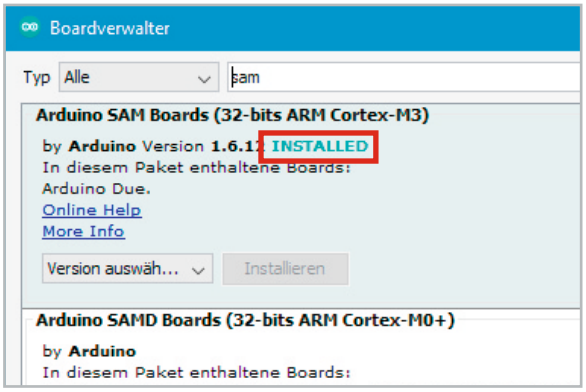


Bild 3: Erfolgreiche Installation des Pakets

Schritt 3

STM32duino installieren

Im nächsten Schritt wird die offizielle STM32duino-Erweiterung von STMicroelectronics installiert. Hierzu wird in der Arduino IDE bei Datei → Voreinstellungen → Zusätzliche Boardverwalter-URLs folgende Adresse eingetragen:

https://github.com/stm32duino/BoardManagerFiles/raw/master/STM32/package_stm_index.json

Außerdem werden bei „Ausführliche Ausgabe während" Häkchen bei Kompilierung und Hochladen gesetzt (Bild 4). Anschließend wird der Dialog mit OK beendet.

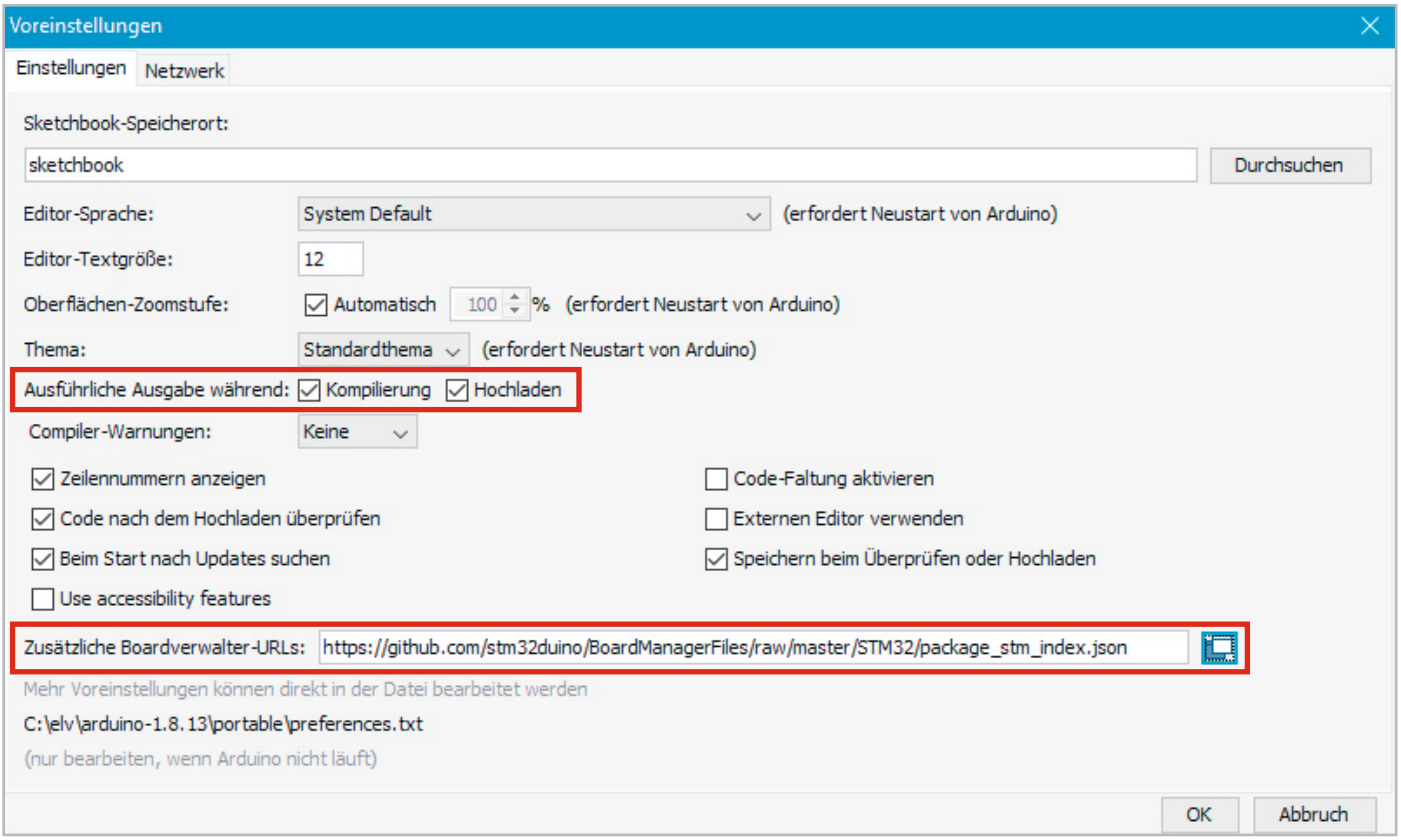


Bild 4: Eintragen der zusätzlichen Boardverwalter-URL

Unter Werkzeuge → Board → Boardverwalter wird oben in das Suchfeld „STM32 Cores“ eingegeben. Dadurch sollte das Board „STM32 Cores by STMicroelectronics“ gefunden werden (Bild 5). Falls nicht, funktioniert vermutlich der Internetzugriff aus der Arduino IDE heraus nicht. Gegebenenfalls muss dann unter Datei → Voreinstellungen → Netzwerk die Proxy-Konfiguration vorgenommen werden.

Auch hier dauert die Installation einige Zeit. Wenn der Hinweis „Installed“ erscheint, wurde der „Arduino core support for STM32 based boards“ installiert. Unter Werkzeuge → Board erscheint nun ein weiterer Eintrag „STM32 Boards“ (Bild 6).

Schritt 4

Arduino_STM32 installieren

Als Nächstes wird der Arduino_STM32-Core von Roger Clark installiert. Dieser wird benötigt, um später die alternative Oszilloskop-Firmware DLO-138 kompilieren zu können. Zuerst wird die Arduino IDE beendet. Im Ordner „portable“ wird im Unterordner „sketchbook“ ein neuer Ordner „hardware“ angelegt (falls noch nicht vorhanden). Auf der Seite [9] wird das Zip-File heruntergeladen und entpackt.

Im entpackten Ordner Arduino_STM32-master befindet sich ein weiterer Ordner Arduino_STM32-master, der in Arduino_STM32 umbenannt wird. Anschließend wird Arduino_STM32 in den angelegten Ordner „hardware“ verschoben. Im Beispiel liegen die Dateien dann im Verzeichnis:

C:\elv\arduino-1.8.13\portable\sketchbook\hardware\Arduino_STM32 (Bild 7)

Schritt 5

Nun wird noch der Windows-Treiber installiert, um später den USB-Bootloader für den Upload der Sketches

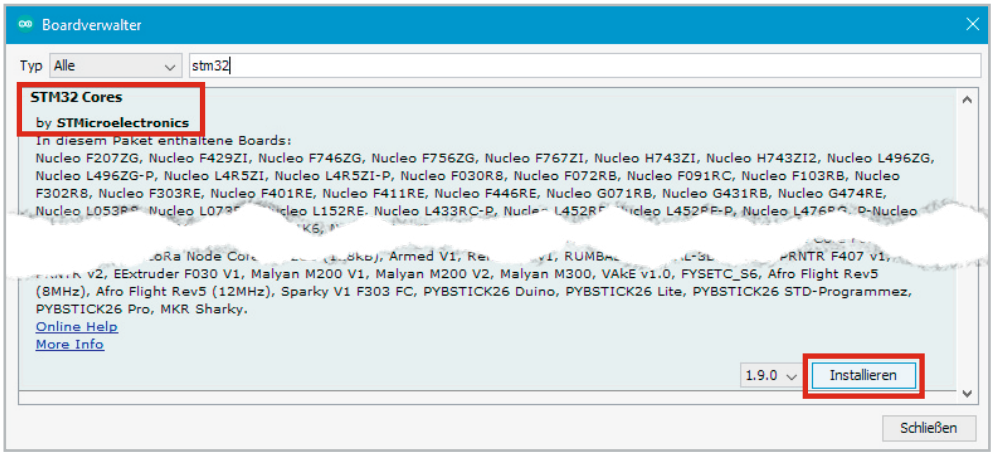


Bild 5: Paket STM32-Cores

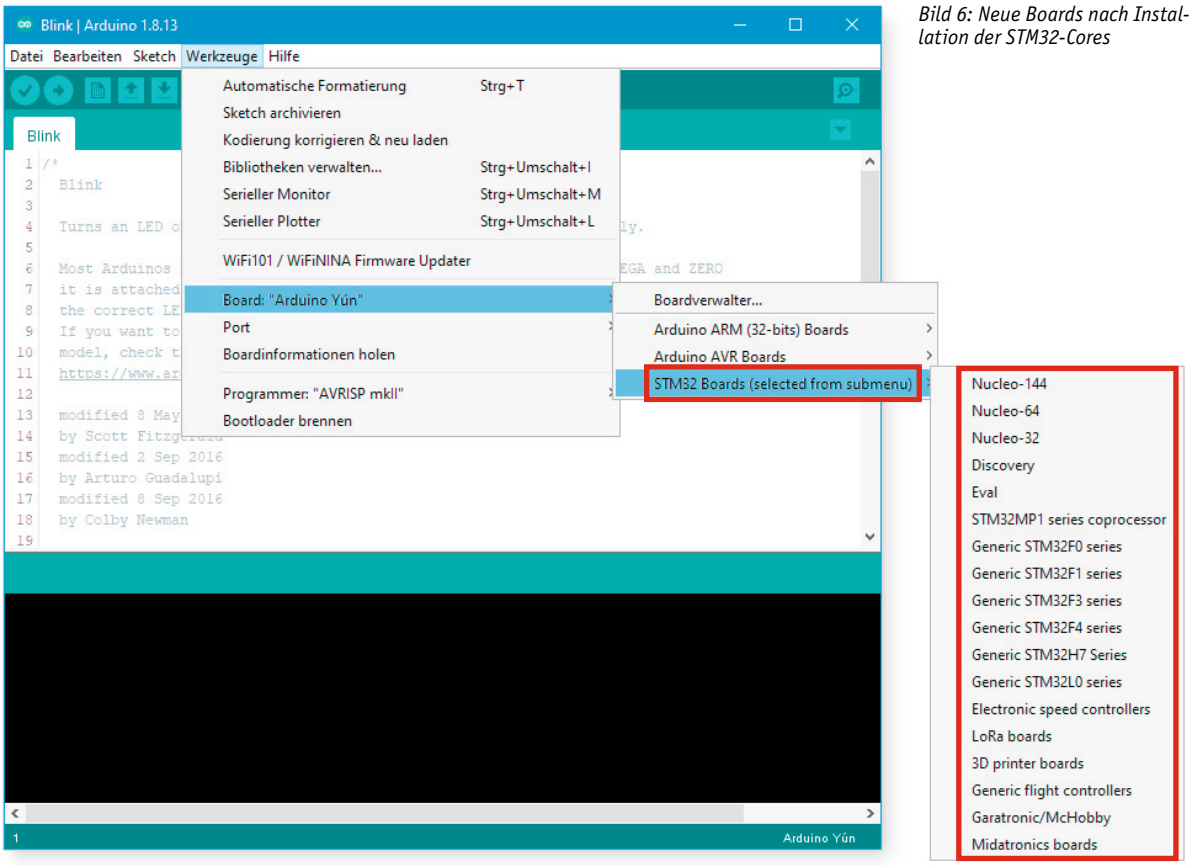


Bild 6: Neue Boards nach Installation der STM32-Cores

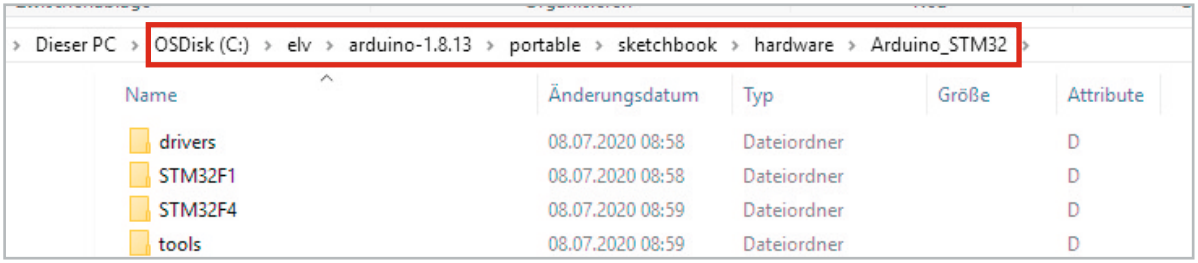


Bild 7: Verzeichnisstruktur mit Arduino_STM32-Core

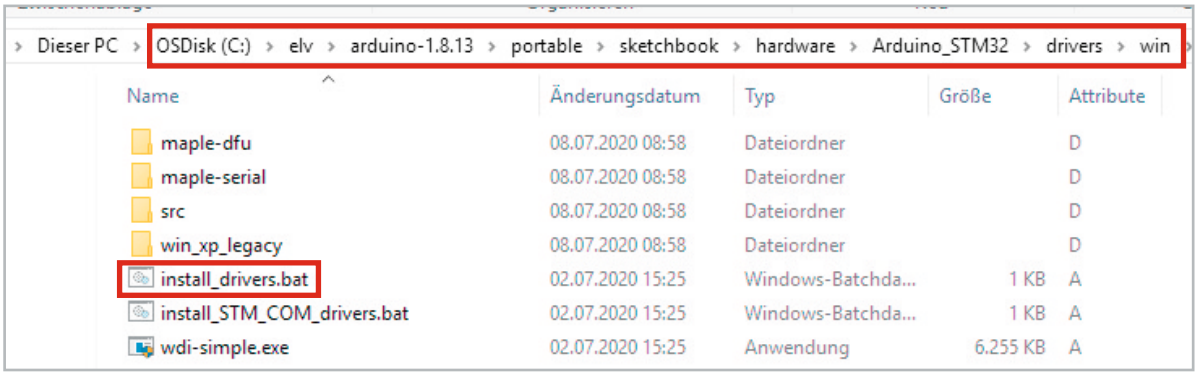


Bild 8: Batchdatei für Windows-Treiber

nutzen zu können. Im Unterordner drivers\win von Arduino_STM32 wird hierzu die Batchdatei install_drivers.bat ausgeführt (Bild 8).

Dadurch werden der „Maple DFU driver" sowie der „Maple serial driver" installiert (Bild 9).

Nach dem Start der Arduino IDE sollte unter Werkzeuge → Board der Eintrag „STM32F1 Boards (STM32duino.com)" vorhanden sein (Bild 10).

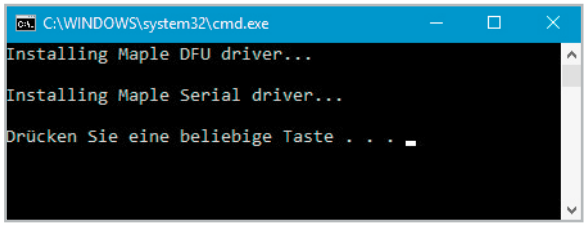


Bild 9: Installation der Maple-Treiber

Schritt 6

STM32CubeProgrammer installieren

Um einen Programmcode in einen fabrikneuen STM32-Mikrocontroller mit gelöschtem Flashspeicher schreiben zu können, wird das Tool „STM32CubeProgrammer" auf dem Zielsystem installiert, den es leider nicht als portable Version gibt. Die Installationsdatei kann unter [10] heruntergeladen werden. Hierzu klickt man dort auf „Get Software" und gibt nach dem Akzeptieren der Lizenzbedingungen die eigene E-Mail-Adresse an, dann erhält man einen Download-Link.

Im User-Manual ist im Kapitel 1.2 [11] der Installationsvorgang kurz erwähnt.

Während des Installationsvorgangs kann man das optional angebotene Tool „STM32 trusted package creator" überspringen, da dieses hier nicht benötigt wird. Es sollten jedoch alle USB-Treiber installiert werden, die angeboten werden.

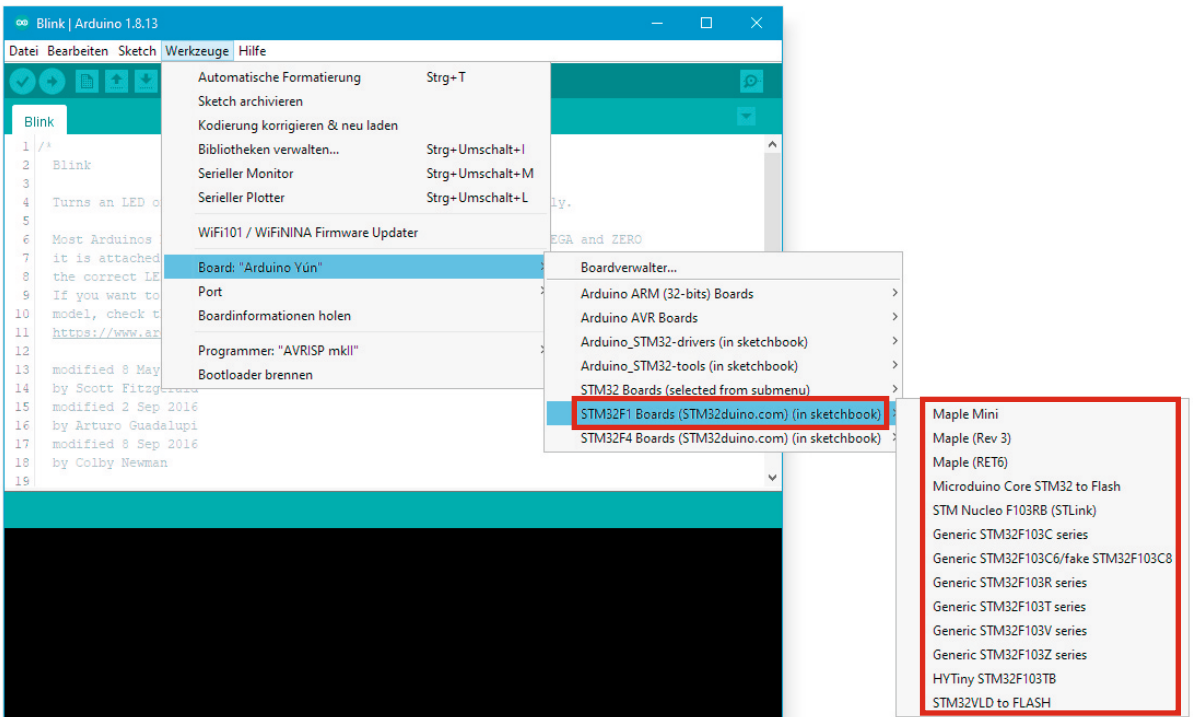


Bild 10: Neue STM32-Boards im Board-Menü

Schritt 7

Alternative Oszilloskop-Firmware DLO-138

Für das DSO138 ist eine alternative Firmware namens DLO-138 verfügbar, die mehr Features bietet. Der Quellcode steht unter [12] bereit.

Die Speichertiefe wurde auf 2048 Samples verdoppelt. Anstatt der Plus-, Minus- und Select-Taste kann ein Drehen-Coder (z. B. KY-040) angeschlossen werden, um eine komfortablere Bedienung zu ermöglichen. Die Firmware unterstützt auch einen zweiten Analogkanal, wobei hierfür dann die entsprechende Mess-Hardware nachgerüstet werden muss, entsprechend der im Schaltplan ersichtlichen Schaltung des ersten Analogkanals. Ohne Hardwaremodifikation sind jedoch zwei zusätzliche Digitalkanäle möglich, die bereits auf der Stiftleiste J6 (SWD-Port) herausgeführt sind.

Ein sehr nützliches Feature ist auch die Ausgabe der aufgezeichneten Daten im CSV-Format auf der seriellen Schnittstelle. Dies kann entweder direkt per UART über die Stiftleiste J5 mithilfe eines dort angeschlossenen USB-UART-Moduls erfolgen oder aber noch komfortabler über die USB-Buchse des DSO138, die als USB-CDC-Schnittstelle einen virtuellen COM-Port auf dem PC erzeugt. Die ausgelesenen Daten können beispielsweise in Libreoffice oder Excel importiert und entsprechend ausgewertet oder grafisch dargestellt werden. Ein Nachteil der DLO-138-Firmware soll nicht verschwiegen werden: Der 10µs/div-Bereich ist nicht mehr möglich.

Schritt 8

DLO-138-Firmware kompilieren

Um die DLO-138 Firmware mit der Arduino IDE zu kompilieren, wird zunächst der Quellcode als Zip-Datei unter [13] heruntergeladen.

Im entpackten Ordner dlo-138-master gibt es einen weiteren Ordner dlo-138-master, dieser muss in dlo-138 umbenannt werden. Darin gibt es den Sketch DLO-138.ino, der über die zuvor installierte Arduino-Portable-Version geöffnet wird.

Tipp:
In der Setup-Funktion sorgt die Anweisung

```
afio_cfg_debug_ports(AFIO_DEBUG_NONE);
```

dafür, dass die Debug-Pins (PA13, PA14, PA15, PB3 and PB4) für JTAG und SWD des STM32 bei der aktuellen Version von Arduino_STM32 als normale GPIOs verfügbar sind. Andernfalls könnte das Display nicht richtig initialisiert werden und die Nutzung der zusätzlichen Digitalkanäle wäre nicht möglich.

In der Datei global.h wird die Zeile 2 durch zwei Schrägstriche auskommentiert, da zunächst die Bedienung über die bereits vorhandenen Tasten erfolgen soll (Bild 11):

```
//#define USE_ENCODER
```

Dadurch erfolgt die Bedienung wie im Original über die vier Tasten. Im zweiten Teil dieses Beitrags im kommenden ELVjournal wird gezeigt, wie sich ein Drehen-Coder nachrüsten lässt.

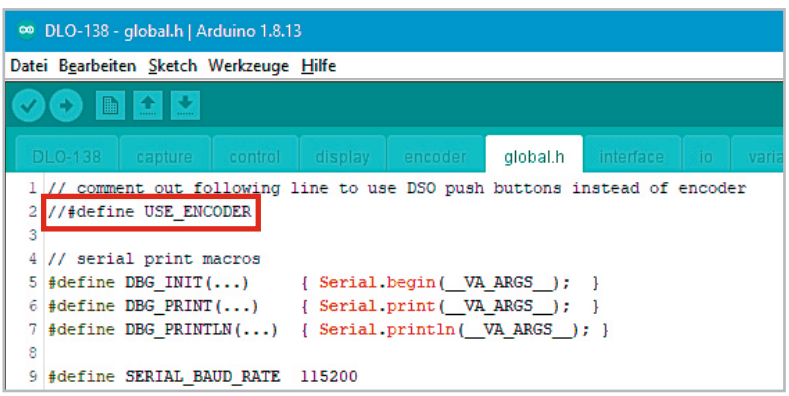


Bild 11: Auskommentieren der Encoder-Nutzung

Schritt 9

Im nächsten Schritt muss noch eine Arduino-Bibliothek für die Grafikausgabe auf dem TFT installiert werden. Hierzu navigiert man zu Sketch → Bibliothek einbinden → Bibliotheken verwalten und gibt im Suchfenster "Adafruit GFX Library" ein (Bild 12).

Achtung: Es darf nicht die neueste Version installiert werden, da es ansonsten zu Compilerfehlermeldungen kommt. Stattdessen muss die Version 1.7.5 ausgewählt und installiert werden.

Die Meldung, dass weitere Bibliotheken benötigt werden, quittieren wir mit „Install Adafruit GFX Library only".

Schritt 10

Anschließend muss noch die passende Board-Konfiguration eingestellt werden. Unter Werkzeuge → Board wird bei „STM32F1 Boards (STM32duino.com)" der Eintrag „Generic STM32F103C series" unter Werkzeuge → Variant „STM32F103C8 (20 k RAM, 64 k Flash)" ausgewählt. Außerdem selektiert man unter Werkzeuge → Upload method den Eintrag „STM32duino Bootloader". Der passende Port wird später eingestellt (Bild 13).

Schritt 11

Nun kann der Sketch kompiliert werden (Menü Sketch → Überprüfen/Kompilieren). Hierbei treten folgende Compiler-Warnings auf, die ignoriert werden können:

- Warning: register range not in ascending order
- Wenig Arbeitsspeicher verfügbar, es können Stabilitätsprobleme auftreten

Falls sonstige Compiler-Fehlermeldungen auftreten sollten, wurde möglicherweise einer der zuvor beschriebenen Installationsschritte nicht ausgeführt oder das falsche Board ausgewählt bzw. die falsche Bibliotheksversion der Adafruit GFX Library installiert.

Schritt 12

Firmware-Version des DSO138 notieren

Unbedingt beachten!
Bevor die Original-Firmware auf dem DSO138 überschrieben wird, sollte die auf dem Startbildschirm des Oszilloskops angezeigte Firmware-Version notiert werden, beispielsweise 113-13801-060, wobei vor allem die letzten drei Stellen relevant sind. Auf der Seite des Herstellers [14] stehen die verfügbaren Versionsstände bereit. Hier sollte man sich vergewissern, dass die auf dem DSO138 installierte Version gelistet ist, um diese bei Bedarf später wieder einspielen zu können.

Wenn die Version kleiner als 050 ist, dann handelt es sich vermutlich um einen alten Hardware-Stand, bei dem der Widerstand R11 mit 1,5 kΩ bestückt ist anstatt mit 150 Ω wie bei den neueren Hardware-Versionen. In diesem Fall sollte der Widerstand getauscht werden, da ansonsten das angezeigte Messsignal bei der DLO-138-Firmware um ca. Faktor 3 zu klein ist. Die Original-Firmware setzt ab Version 050 ebenfalls einen Wert von 150 Ω voraus.

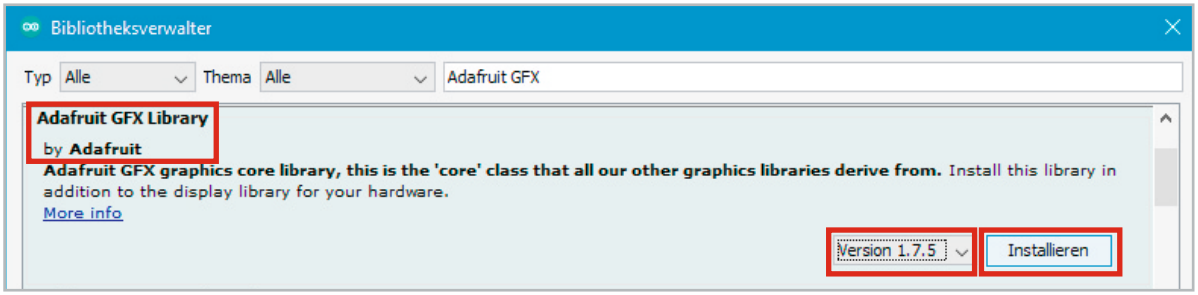


Bild 12: Installation der Adafruit GFX Library in der Version 1.7.5

Schritt 13

Gehäuse öffnen

Hinweis: Das Öffnen des Gehäuses und die Modifikation der Platine erfolgen auf eigenes Risiko und eigene Gefahr. Außerdem erlöschen sämtliche Garantieansprüche.

Um das Plexiglasgehäuse zu öffnen, schraubt man zuerst die vier Hutmuttern an der Oberseite ab und entfernt die obere Platte. Anschließend werden die vier Seitenteile herausgezogen. Danach zieht man die komplette Einheit, die aus Platine, den drei Plexiglasplatten sowie den Schalter- und Tastkappen besteht, nach oben heraus und legt diese Einheit auf den Tisch, sodass das TFT-Display nach oben zeigt. Nun kann die Grundplatte kopfüber mit den Schrauben von oben durch die Einheit gesteckt werden, sodass sich die Grundplatte oberhalb des Displays befindet und auf den Schalter- und Tastkappen aufliegt. Dadurch lässt sich nun der Aufbau bequem umdrehen, sodass die Lötseite der Platine zugänglich ist. Auf diese Weise kann man beispielsweise das für den Bootloader benötigte Kabel anlöten oder die Litzen für das später beschriebene Encodermodul.

Die Platine wird umgedreht in das Gehäuse gelegt, sodass die Lötseite nach oben zeigt (Bild 14, oben).

Schritt 14

DLO-138-Firmware auf DSO138 laden

Um den kompilierten DLO-138-Sketch auf das DSO138-Board zu laden, muss zunächst der passende Bootloader in den STM32-Controller geflasht werden. Hierzu wird der von STMicroelectronics eingebaute Bootloader im STM32F103 verwendet, der über die USART-Pins PA9 (TxD-Ausgang) und PA10 (RxD-Eingang) kommuniziert. Praktischerweise sind diese beiden Pins zusammen mit GND auf dem Steckverbinder J5 herausgeführt.

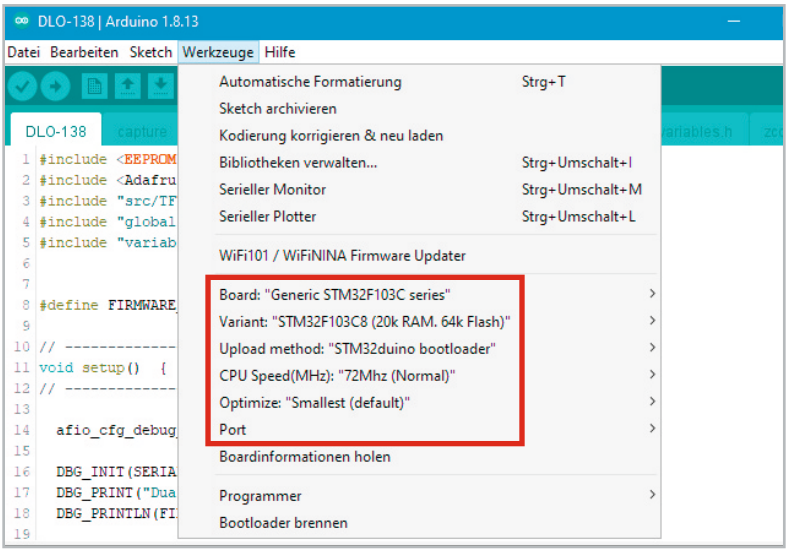


Bild 13: Boardkonfiguration

Um den Bootloader-Modus zu aktivieren, muss der BOOT0-Pin des STM32 beim Booten auf 3,3 V Highpegel gelegt werden. Dieser Pin ist auf der Lötbrücke JP1 auf der Platinenunterseite sowie dem danebenliegenden Testpunkt zugänglich. Hier wird das abgeschnittene Ende eines Dupont-Kabels angelötet, sodass das andere Ende mit Buchse auf die Stiftleiste J6 Pin1 (3,3 V) gesteckt werden und so den Bootloadermodus aktivieren kann. Auf dem Foto ist es das orange Kabel.

Danach wird die Platine vorsichtig wieder umgedreht und die Gehäuseunterseite wieder von unten eingesteckt.

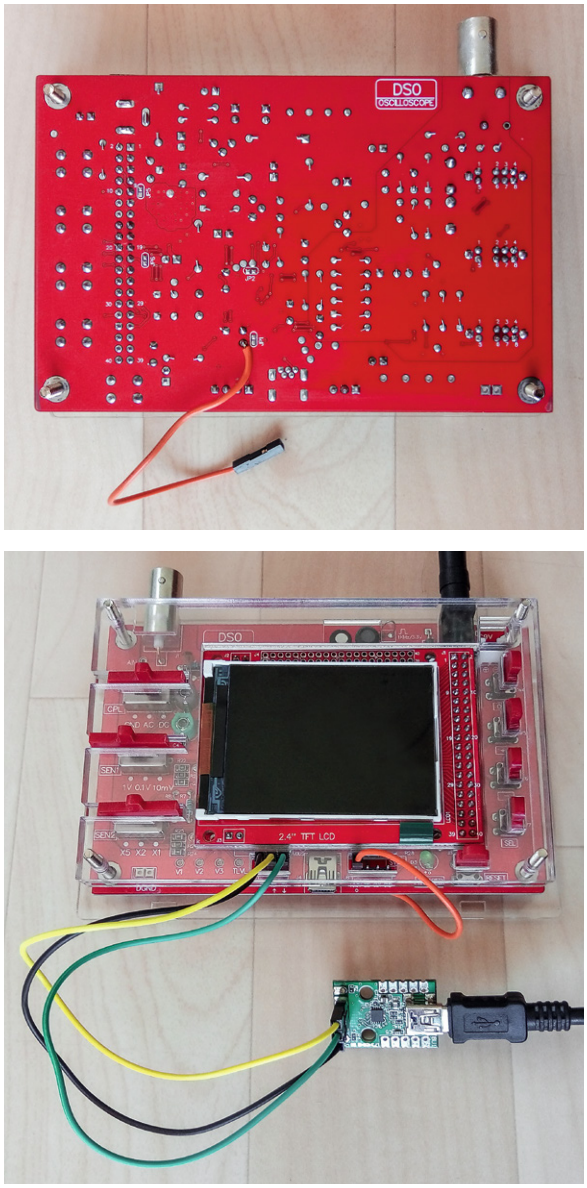


Bild 14: Nach oben zeigende Lötseite (oben) und Verkabelung der seriellen Schnittstelle (unten)

Um die USART-Pins am PC anschließen zu können, benötigt man ein USB-UART-Modul mit 3,3-V-Pegeln, beispielsweise das UM2102N von ELV [15]. Dort lötet man an ST1 eine dreipolige Stiftleiste an den Signalen GND, RxD und TxD an, sodass die Verbindung zum DSO138 über drei Dupont-Kabel des Typs Buchse-Buchse vorgenommen werden kann, z. B. [16]. TxD muss dabei mit RxD der Gegenstelle und umgekehrt RxD mit TxD verbunden werden.

Auf Bild 14, unten verbindet das schwarze Kabel jeweils GND, das gelbe Kabel RxD des DSO138 mit TxD des UM2102N und das grüne Kabel TxD des DSO138 mit RxD des UM2102N.

Der Bootloader wird auf [17] bereitgestellt. Hierzu lädt man die Binärdatei [18] auf den PC herunter. Neben dem Bootloader enthält diese auch gleich eine zyklische Ausgabe auf dem USB-CDC-Port, sodass man später leicht kontrollieren kann, ob der Bootloader erfolgreich installiert wurde.

Als Flash-Tool auf dem PC wird der zuvor installierte STM32CubeProgrammer verwendet. Nach der passenden Verkabelung mit den vier Dupont-Kabeln wird anschließend die Spannungsversorgung des DSO138 über das Steckernetzteil an J10 eingeschaltet.

Nach dem Start des Tools müssen zuerst die UART-Verbindungseinstellungen vorgenommen werden. Hierzu wählt man rechts im blauen Feld „UART" aus sowie bei „Port" die passende virtuelle COM-Schnittstelle des USB-UART-Moduls, alle übrigen Einstellungen bleiben unverändert. Im Beispiel wird COM14 verwendet.

Um die Verbindung herzustellen, wird kurz der Reset-Taster des DSO138 betätigt und danach auf die Schaltfläche „Connect" geklickt. Falls die Meldung „Error: Activating device: KO" erscheint, kontrolliert man noch mal alle Verbindungen und die richtige COM-Port-Einstellung und versucht es nach nochmaligem Drücken des Reset-Tasters erneut.

Bei erfolgreicher Verbindung erscheint beim DSO138 von JOY-iT folgende Meldung, da im Auslieferzustand der Ausleseschutz des STM32-Controllers aktiv ist (Bild 15).

Es ist dann nicht möglich, die Original-Firmware auszulesen und zu sichern (s. Hinweis oben). Der Ausleseschutz kann über das Tool entfernt werden, allerdings wird dadurch zwingend automatisch der komplette Flash gelöscht.

Bitte daher unbedingt wie oben beschrieben die vorhandene Firmware-Version notieren!

Um dies durchzuführen wird die Protection-Meldung mit OK bestätigt. Kurz darauf erscheint der Hinweis, dass der Ausleseschutz erfolgreich entfernt wurde (Bild 16). Anschließend muss erneut auf den Connect-Button geklickt werden.

Tipp: Falls im Auslieferzustand kein Ausleseschutz vorhanden sein sollte, kann die Originalfirmware ausgelesen werden. Hierzu wählt man unter „Memory & File edition" als Adresse 0x08000000 und als Size 0x20000 aus, wählt mit dem Pfeil rechts im Read-Button „Save as" aus und gibt als Dateinamen dso138_original.hex ein. Nun sollte die angegebene Hex-Datei als Sicherungskopie der Originalfirmware angelegt worden sein.

Im Dialog „Erasing & Programming" („Burger"-Menü oben links) wird nun die zuvor heruntergeladene Bootloaderdatei dso138_boot20.bin ausge-

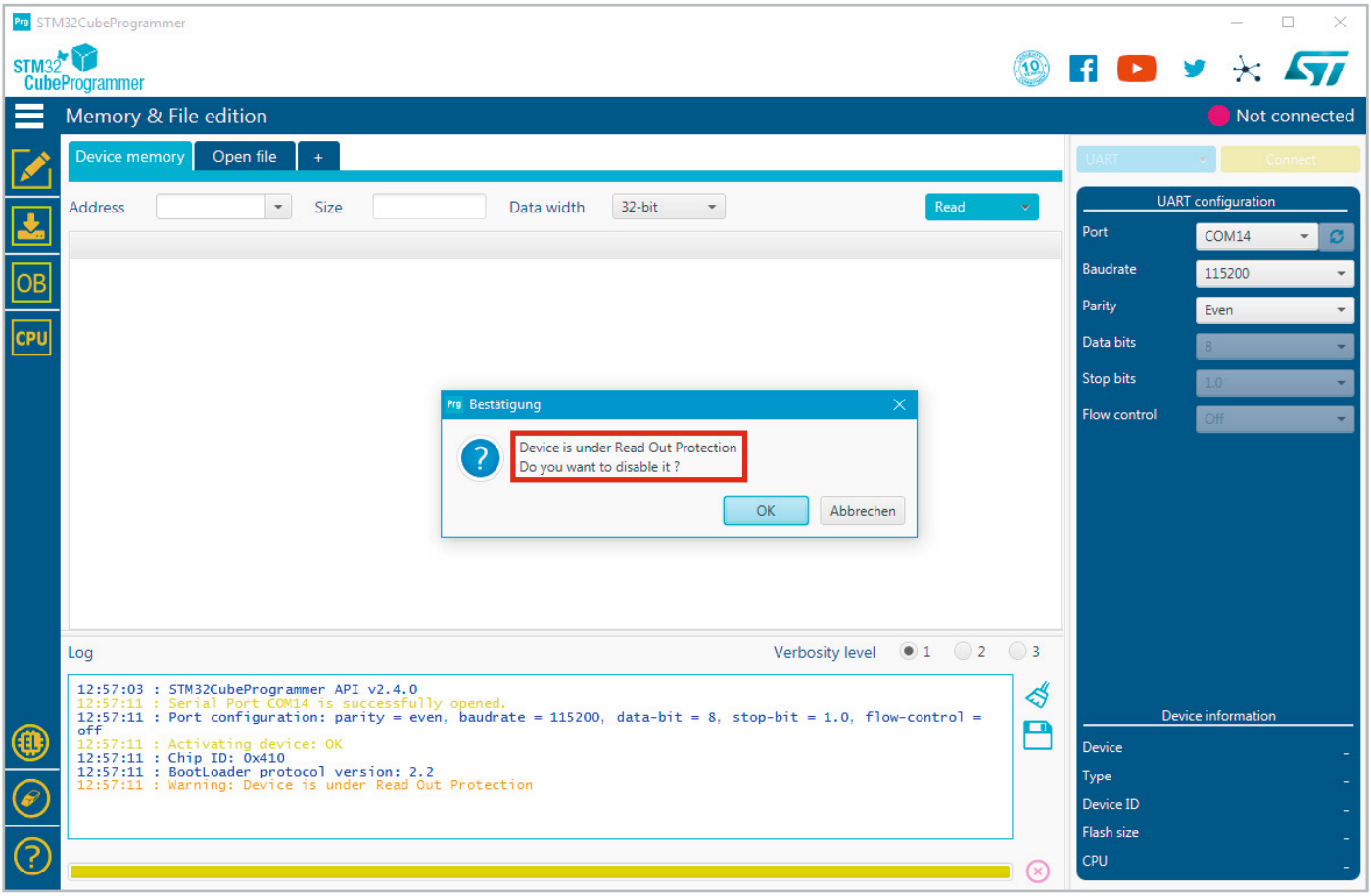


Bild 15: Entfernung des Flash-Speicher-Ausleseschutzes

wählt – ein Häkchen bei „Verify Programming" setzen – und durch Klick auf „Start Programming" in den STM32 geflasht (Bild 17).

Nach erfolgreicher Programmierung wird auf „Disconnect" geklickt und das Tool beendet. Die Spannungsversorgung wird nun vom DSO138 getrennt und das orange Kabel vom Vcc-Pin abgezogen. Das USB-UART-Modul kann, wenn keine Daten über USART1 (z. B. über die Arduino-IDE, s. u.) abgerufen werden sollen, ebenfalls abgesteckt werden.

Schritt 15

Nach dem Wiedereinschalten der Spannungsversorgung (es wird zunächst nur ein weißer Bildschirm angezeigt!) wird ein USB-Kabel in die Mini-USB-Buchse des DSO138 gesteckt und mit dem PC verbunden. Dadurch sollte Windows das Gerät erkennen und automatisch den Maple-Treiber installieren. Im Geräte-Manager sollte dann auch ein weiterer virtueller COM-Port „Maple Serial" vorhanden sein. Hier im Beispiel ist dies COM27 (Bild 18).

Nun wird die Arduino IDE gestartet und der Maple-Serial-Port in der Arduino IDE unter Werkzeuge → Port ausgewählt. Durch Klick auf die Lupe oben rechts wird der serielle Monitor der Arduino IDE gestartet. Dort sollten nun entsprechende Begrüßungsmeldungen sichtbar sein als Zeichen dafür, dass der Bootloader korrekt installiert wurde (Bild 19).

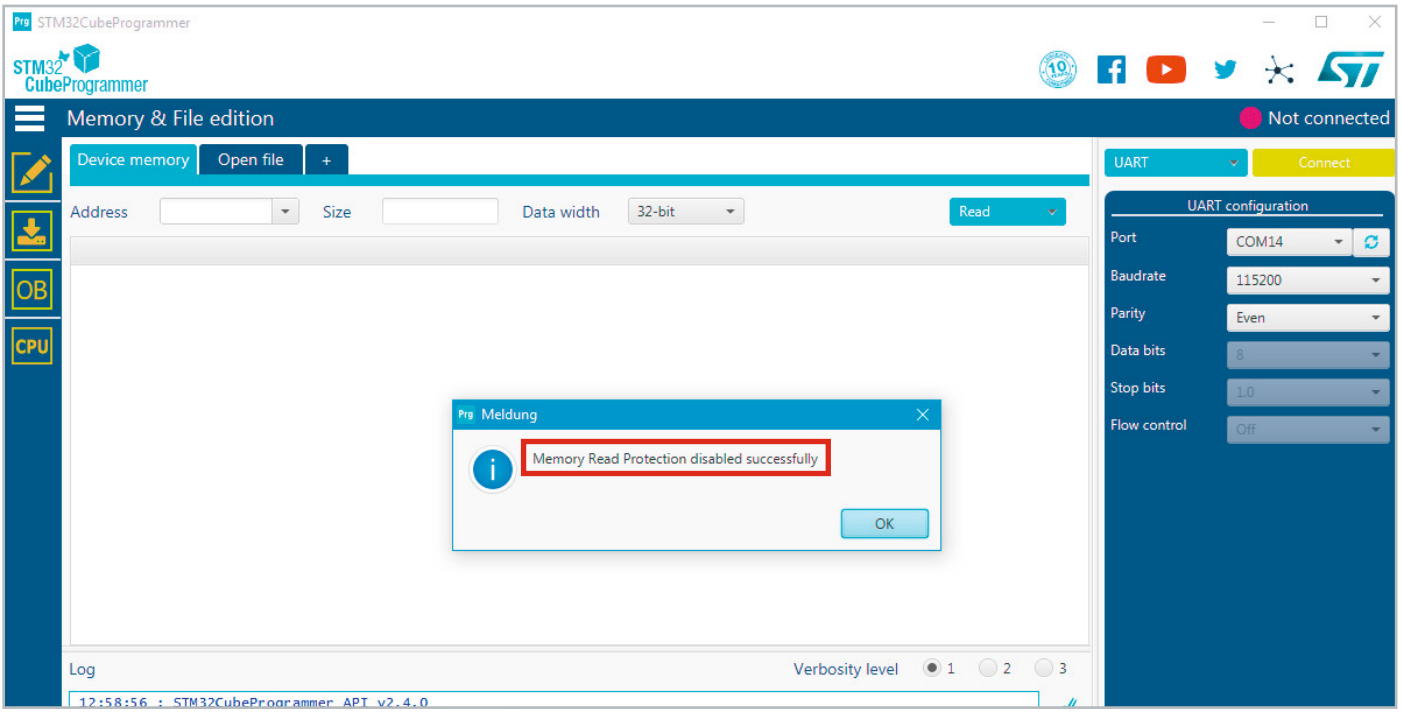


Bild 16: Deaktivierter Ausleseschutz

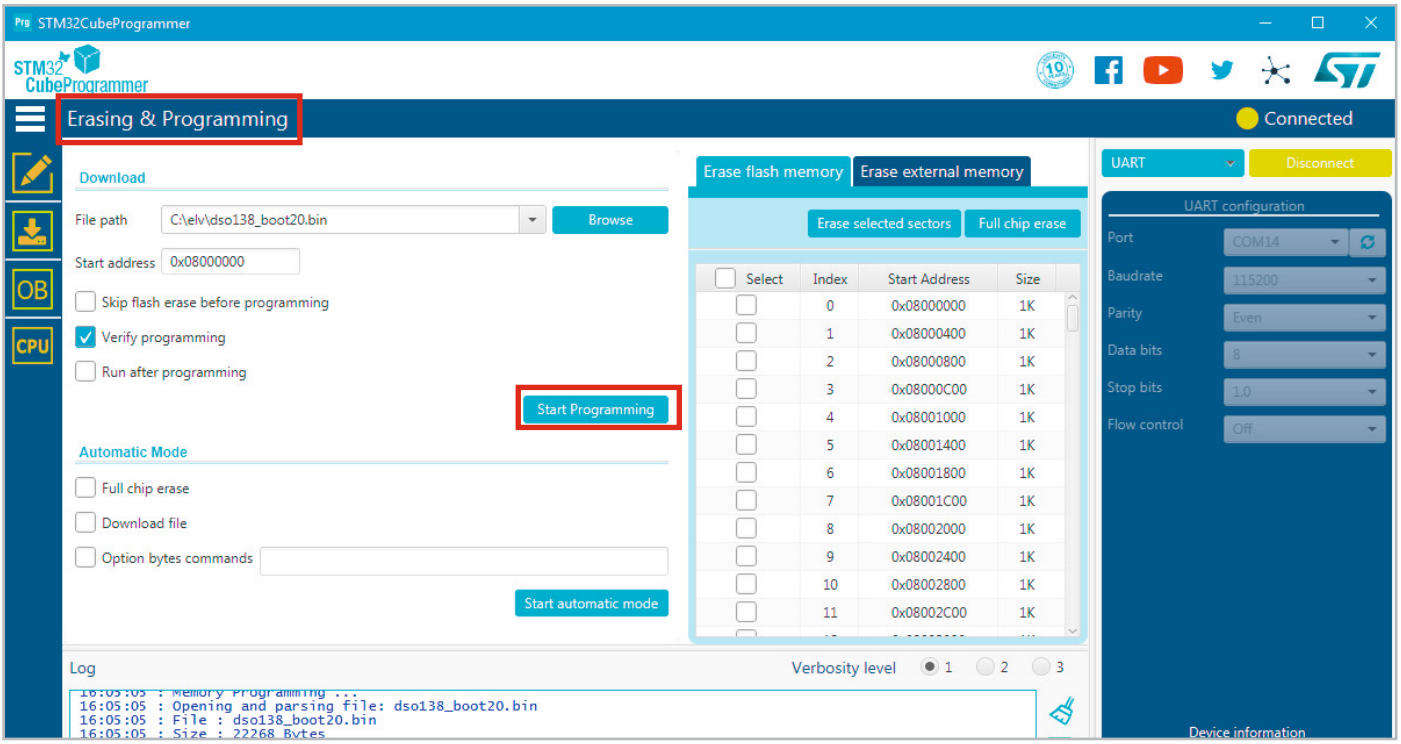


Bild 17: Flashen des Bootloaders

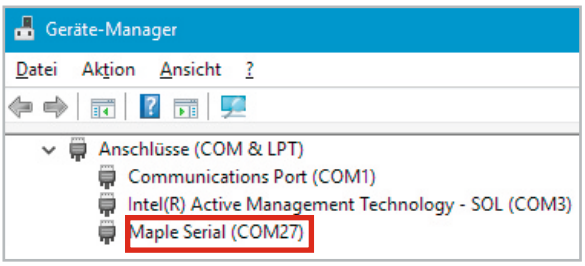


Bild 18: DSO138 verbunden über den Maple-Serial-Treiber

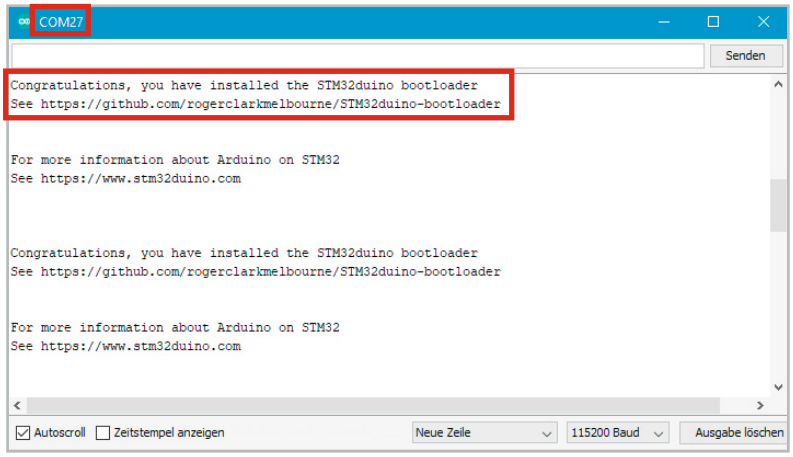


Bild 19: Ausgabe im seriellen Monitor der Arduino IDE

Schritt 16

Nun ist es endlich so weit: Die DLO-138-Firmware kann auf das DSO138 geladen werden. Nach Klick auf Sketch → Hochladen sollten entsprechende Meldungen des Bootloaders im unteren Ausgabefenster der Arduino IDE erscheinen, wobei die Meldung „Error resetting after download" ignoriert werden kann.

Falls nach der Meldung „Searching for DFU device" dieses nicht automatisch gefunden wird, kann kurz die Reset-Taste des DSO138-Boards gedrückt werden.

Auf dem DSO138-Board sollte nun kurz der Begrüßungsbildschirm der DLO-138-Firmware zu sehen sein (Bild 20) und anschließend die Messsignale.

Tipp: Falls nicht, fehlt evtl. die Zeile zum Abschalten der Debug-Pins in der Setup-Funktion (s. o.):

```
afio_cfg_debug_ports(AFIO_DEBUG_NONE);
```

Für einen ersten Test kann die BNC-Buchse über das mitgelieferte Kabel mit dem 3,3-V-Rechteck-Testsignal verbunden werden. Die rote Messklemme muss hierzu an die Drahtschlaufe an J2 geklemmt werden, die schwarze Messklemme bleibt offen (Bild 21).

Falls der Lowpegel nicht auf der Nulllinie liegt, muss ein Abgleich erfolgen wie nachfolgend unter Bedienung beschrieben.

Bedienung der DLO-138-Firmware

Die Bedienung des Oszilloskops mit DLO-138-Firmware erfolgt ähnlich wie bei der Original-Firmware:

Taste	Funktion
Select	setzt den Fokus auf den nächsten Parameter (wird eingerahmt auf Display)
Plus/Minus	verändert den aktuell ausgewählten Parameter
OK (kurzer Tastendruck)	friert die Aufzeichnung der Messdaten ein (HOLD) und gibt die Daten auf der seriellen Schnittstelle aus (USB-CDC). Mit erneutem Tastendruck wird die HOLD-Funktion wieder beendet

Nach einem langen Tastendruck (ca. 3 s) der OK-Taste werden beim Loslassen abhängig vom gerade selektierten Parameter folgende Einstellungen vorgenommen:

Selektierter Parameter	Funktion
Triggerlevel	Triggerlevel für Analogkanal 1 wird genullt
Horizontale Scrollbar	Scrollbar zum Blättern durch die Messdaten wird mittig positioniert
Vertikale Nullposition von Analogkanal 1	Nullposition wird in die Mitte gelegt. Wenn zusätzlich der Schalter für Coupling auf GND steht, erfolgt der Nullabgleich des Messsignals
Alle anderen Parameter	Statistikanzeige für Analogkanal 1 wird ein- bzw. ausgeblendet

Die Ausgabe der Messdaten (Bild 22) erfolgt im CSV-Format, sobald mit kurzem Druck der OK-Taste die Oszilloskop-Anzeige auf „HOLD" gesetzt wird. Auf welcher Schnittstelle die Daten ausgegeben werden, wird durch die Defines DBG_PRINT usw. in global.h festgelegt. „Serial" ist die USB-CDC-Schnittstelle. Falls man die Daten über ein an J5 angeschlossenes

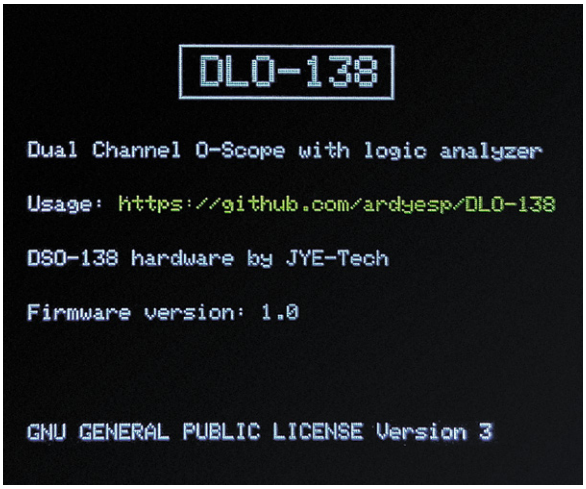


Bild 20: Begrüßungsbildschirm mit neuer Firmware DLO-138

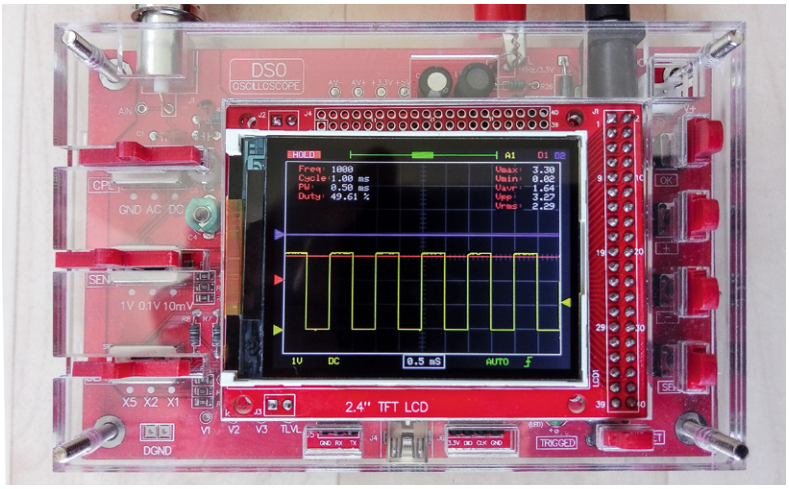


Bild 21: Erster Test mit neuer Firmware

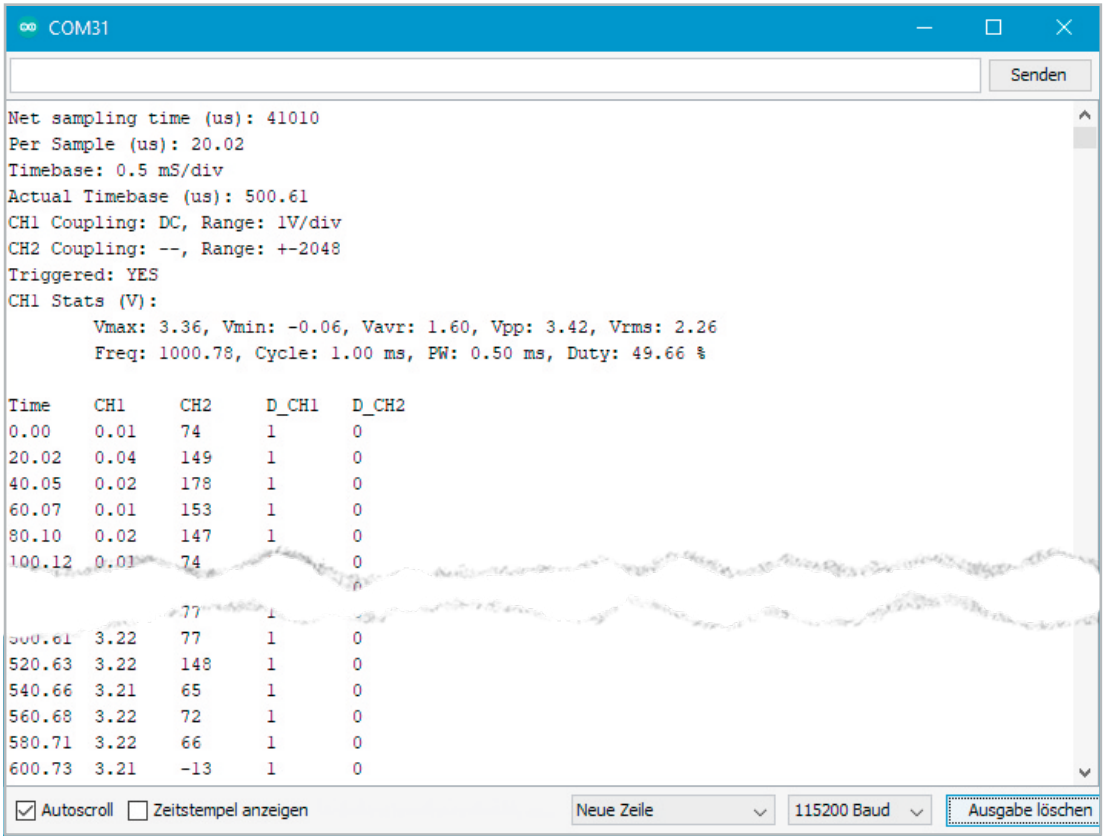


Bild 22: Ausgabe der Messdaten im seriellen Monitor der Arduino IDE

USB-UART-Modul ausgeben möchte, ändert man den Define auf „Serial1" ab. Die Daten können in eine Textdatei kopiert und in OpenLibre oder Excel importiert werden. Beim Import muss das Dezimaltrennzeichen entsprechend auf einen Punkt umgestellt werden.

Abschließend können die Seiten- und die Frontplatte wieder eingebaut, die Hutmuttern verschraubt und das Oszilloskop mit der neuen Firmware genutzt werden.

Ausblick

Hiermit ist die Programmierung des DSO138 mit der neuen Firmware abgeschlossen, und wir werfen im kommenden ELVjournal einen Blick auf mögliche Hardware-Umbauten. Zudem schauen wir uns die Möglichkeiten zur Erstellung eigener Firmware für STM32- und STM8-Controller in der Arduino IDE an. ELV

Weitere Infos:

[1] JOY-iT Oszilloskop DSO138, Artikel-Nr. 127893
[2] Schaltplan DSO138: https://files2.elv.com/public/12/1278/127893/Internet/127893_oszilloskop_schaltplan.pdf
[3] Reference Manual STM32F1-Familie: https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf
[4] Maple-Board von Leaflabs: https://www.leaflabs.com/maple
[5] STMicroelectronics STM32duino-Projekt: https://github.com/stm32duino/Arduino_Core_STM32
[6] Arduino_STM32 von Roger Clark: https://github.com/rogerclarkmelbourne/Arduino_STM32/
[7] Unterschiede Arduino_STM32/ STM32duino-Core: https://github.com/rogerclarkmelbourne/Arduino_STM32/tree/master/STM32F1/system/libmaple/stm32f1/include/series
[8] Aktuelle Version Arduino IDE: https://www.arduino.cc/en/Main/Software
[9] Arduino_STM32-Core: https://github.com/rogerclarkmelbourne/Arduino_STM32/archive/master.zip
[10] STM32CubeProgrammer: https://www.st.com/en/development-tools/stm32cubeprog.html
[11] STM32CubeProgrammer-Manual: http://www.st.com/resource/en/user_manual/dm00403500-stm32cubeprogrammer-software-description-stmicroelectronics.pdf
[12] Alternative Firmware DLO-138: https://github.com/ardyesp/DLO-138
[13] DLO-138-Firmware: https://github.com/ardyesp/DLO-138/archive/master.zip
[14] DSO138-Firmware-Versionen: https://jyetech.com/firmware-dso-138/
[15] USB-UART-Modul UM2102N: Artikel-Nr.: 150952
[16] Dupont-Kabel Buchse-Buchse: https://de.elv.com/velleman-steckbruecken-set-buchse-auf-buchse-15-cm-10-teilig-129020
[17] STM32duino-Bootloader: https://github.com/rogerclarkmelbourne/STM32duino-bootloader
[18] Binärdatei Bootloader: https://github.com/rogerclarkmelbourne/STM32duino-bootloader/blob/master/binaries/dso138_boot20.bin

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