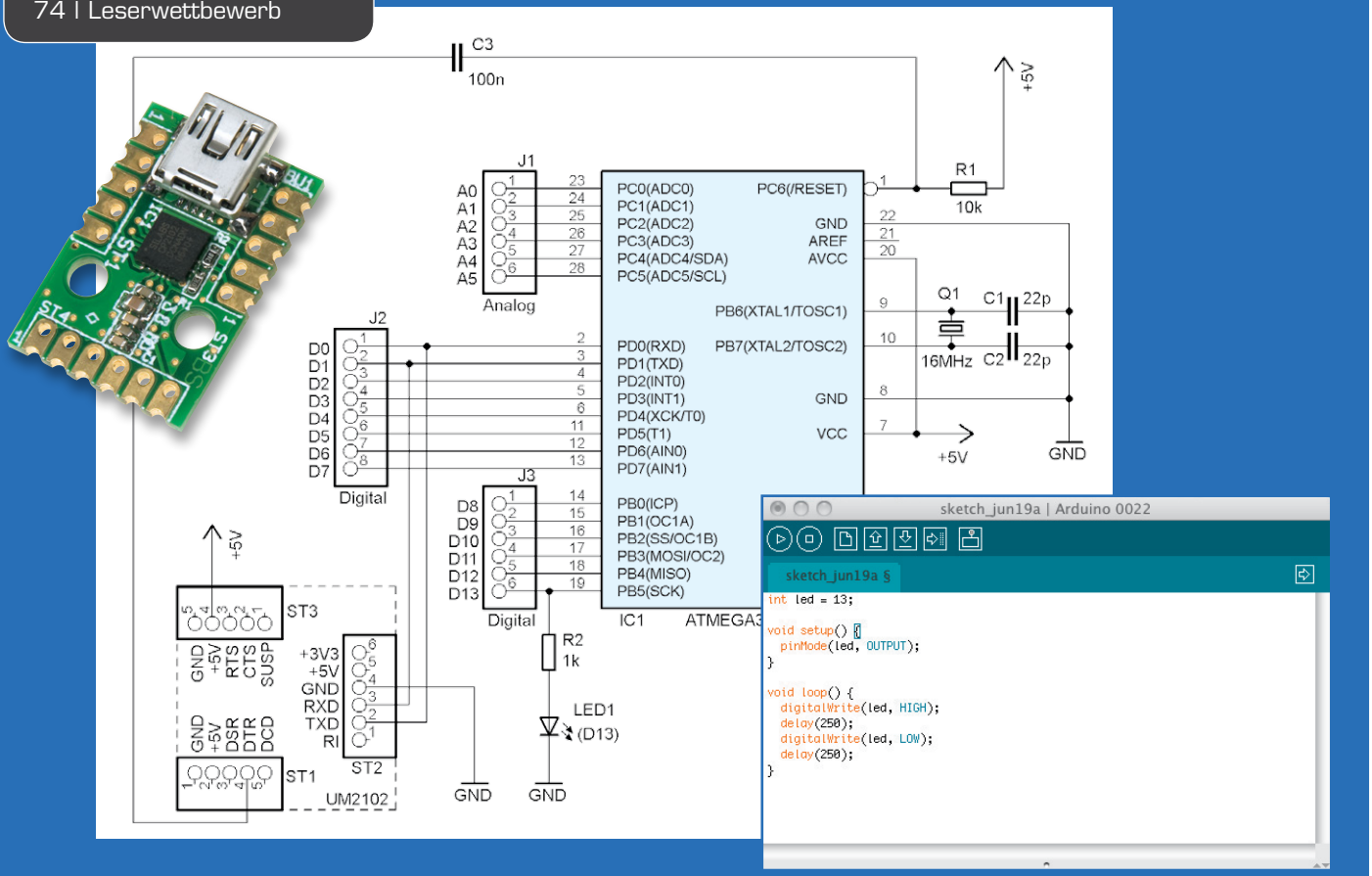

„ELVuino“ – ein einfach aufzubauendes µC-System

Im Rahmen unseres Leserwettbewerbs* hat Herr Gaus aus Schramberg unter Einsatz des USB-Moduls ELV UM2102 eine sehr einfach realisierbare Mikrocontroller-Plattform entwickelt, die durch das Bootloader-Prinzip softwarekompatibel mit dem Arduino-System und entsprechend einfach über dessen IDE programmierbar ist.

Low-Cost-System im Eigenbau

Die Mikrocontroller-Systeme auf Arduino-Basis sind bei Hobbyelektronikern sehr beliebt, speziell auch bei Einsteigern. Anstatt jedoch für eigene Anwendungen jedes Mal einen fertigen Arduino zu kaufen, wäre es doch viel reizvoller, sich selbst einen aus Standardbauteilen zu bauen, ohne einen bereits vorprogrammierten Controller verwenden zu müssen. Dies ist mit dem von mir „ELVuino“ genannten Minimalsystem möglich, noch dazu recht preiswert.

Die Argumente, eigene Anwendungen auf diese Weise zu realisieren, sind sicher nicht von der Hand zu weisen:

- Low-Cost-System, außer einem unprogrammierten AVR-Controller samt tatsächlich notwendiger Peripherie für die Anwendung und dem einmaligen Aufbau der Programmierplattform entstehen keine Kosten
- Einfach aufbaubar, auch für Anfänger
- Kann lötfrei auf einem Steckbrett aufgebaut werden
- USB-Anschluss vorhanden, ohne dass SMD-Bauteile gelötet werden müssen
- ATmega328P wird in der Schaltung mit dem Bootloader programmiert, was einen zusätzlichen Lerneffekt bringt
- Kein ATmega-Controller mit bereits vorprogrammiertem Bootloader erforderlich, man kann also sofort einen vorhandenen Controller nutzen
- Es wird ein handelsüblicher unprogrammierter ATmega328P verwendet
- Preiswertes Fertigmodul UM2102 von ELV für USB-Anbindung mit 5-V- und 3,3-V-Erzeugung on Board
- ATmega328P kann auf einem Steckbrett mit der Arduino-Entwicklungsumgebung programmiert und anschließend stand alone in eigenen Schaltungen eingesetzt werden

Schaltung und Aufbau

In Bild 1 ist die Schaltung der ELVuino-Plattform zu sehen, die sich tatsächlich auf die nötigsten Bauteile beschränkt. Im Zentrum steht natürlich mit IC1 der ATmega328P als Standard-Controller der Arduino-Plattform. R1 stellt ein Controller-Reset beim Anlegen der Betriebsspannung sicher, um einen definier-

*Alle Infos finden Sie unter www.elvjournal.de: Webcode #1257

ten Anlauf des Mikroprozessors zu gewährleisten. Über C3 wird beim späteren Programmieren des AVR ein Reset-Impuls vor dem Übertragen eines Arduino-Sketches ausgelöst.

Der 16-MHz-Quarz Q1 sowie C1/C2 dienen der externen, quarzstabilen Takterzeugung. Wie beim Original ist in der Schaltung das Herausführen aller Ports auf Stift-/Buchsenleisten angedeutet, diese benötigt man allerdings nur, wenn man die Schaltung als Experimentierplattform nutzen bzw. Shields anschließen will. In der Praxis der eigenen Anwendung sind die Ports ohnehin direkt mit entsprechenden Schaltungsteilen wie z. B. einem Display und Sensoren verbunden. Die zur Standardbestückung einer Arduino-kompatiblen Plattform gehörende LED 1 (Arduino: D13) deutet dies an, sie bildet auch die optische Kontrolle beim Programmieren des AVR bzw. des Bootloaders.

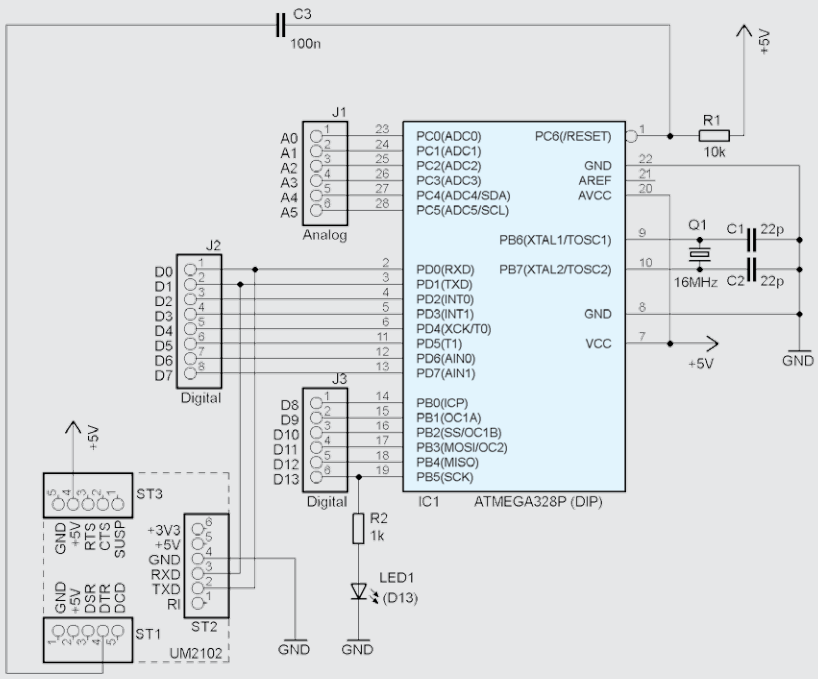

Bild 1: Schaltplan des ELVuino für den normalen Programmier- und Testbetrieb

ST1 bis ST3 stellen die Verbindung zum USB-UART-Baustein UM2102 (Bild 2) her, einem sehr preiswert erhältlichen USB-Baustein von ELV auf Basis des CP2102 von Silicon Labs. Dieser bildet gewissermaßen das „Rückgrat" des kleinen Controllersystems und ist wie alle anderen peripheren Teile auch nur einmal erforderlich.

Die bis hier besprochene Schaltung stellt die normale Beschaltung zum Programmieren des AVR per Arduino-IDE dar. Hierüber kann man also später seine eigenen Programme (Sketches) auf den AVR schreiben und testen.

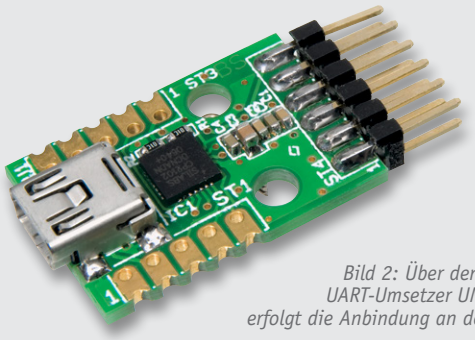

Bild 2: Über den USB-UART-Umsetzer UM2102 erfolgt die Anbindung an den PC.

Programmierung des Bootloaders

Bei einem fabrikneuen Atmega328P muss zunächst einmalig der Flashspeicher mit dem Arduino-Bootloader programmiert werden. Hierfür können einige Steuerleitungen des auf dem ELVuino bereits vorhandenen USB-Seriell-Konverter-Moduls UM2102 [4] verwendet werden, dazu ist die Grundschaltung wie in Bild 3 gezeigt zu modifizieren. Das ist für jeden Controllerchip nur einmal erforderlich, die notwendigen Änderungen sind auf dem Steckbrett schnell ausgeführt.

Folgende Signale müssen für die Programmierung miteinander verbunden werden:

Atmega328P	**UM2102**
Reset (Pin 1)	DTR
SCK (Pin 19)	TXD
MOSI (Pin 17)	RTS
MISO (Pin 18)	CTS

In die Anschlusspins des UM2102 können Präzisions-Sockelstreifen eingelötet werden, sodass dann isolierte Schaltdrähte ein-

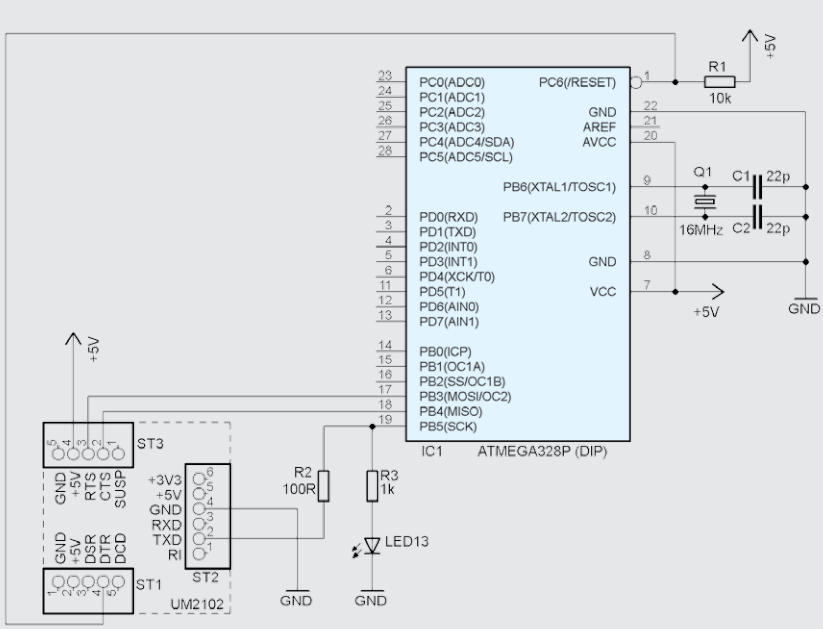

Bild 3: Für das Programmieren des Bootloaders ist die Grundschaltung wie hier gezeigt leicht zu modifizieren.

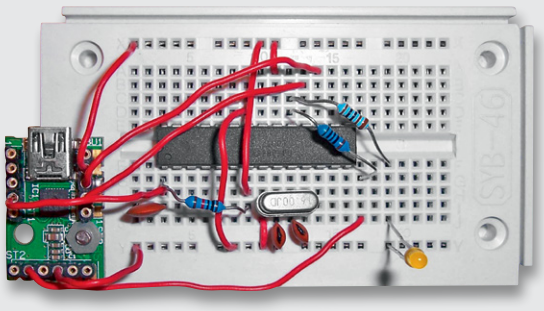

Bild 4: Die Bootloader-Schaltung ist auf einem Steckbrett schnell aufgebaut.

gesteckt und mit dem Steckbrett verbunden werden können. Bild 4 zeigt den Aufbau auf dem Experimentierbrett.

Ein ziemlich universelles Programmiertool, das sehr einfach für diese Programmierung angepasst werden kann, steht mit der bekannten Open-Source-Software AVRDUDE (siehe [1] und [2]) bereit. Dieses Tool wird auch von der Arduino-Entwicklungsumgebung [3] verwendet, sodass die Programmierung des Bootloaders direkt aus der IDE heraus erfolgen kann.

Damit das UM2102 als Programmer unterstützt wird, sind einige Anpassungen notwendig.

```
# ------------------------------------------------------------------------------
# ELVuino
# reset=!dtr sck=!txd mosi=!rts miso=!cts

programmer
  id    = «elvuino»;
  desc  = «Serial AVR Programmer for ELVuino, reset=!dtr sck=!txd mosi=!rts miso=!cts»;
  type  = serbb;
  reset = ~4;
  sck   = ~3;
  mosi  = ~7;
  miso  = ~8;
;
# ------------------------------------------------------------------------------
```

Bild 5: Der Programmteil für die Anpassung der Arduino-IDE

Im Verzeichnis

C:\arduino-1.0.1\hardware\tools\avr\etc

sollte zunächst eine Sicherungskopie von der Konfigurationsdatei *avrdude.conf* erstellt werden, falls beim Editieren etwas schiefgehen sollte.

In die Konfigurationsdatei *avrdude.conf* fügt man dann den in Bild 5 aufgeführten Programm-Abschnitt in den Bereich unterhalb von „# PROGRAMMER DEFINITIONS“ ein. Dadurch wird ein Programmer namens „elvuino“ hinzugefügt, der über den Kommandozeilenparameter *„-c elvuino“* beim Aufruf von AVRDUDE ausgewählt werden kann.

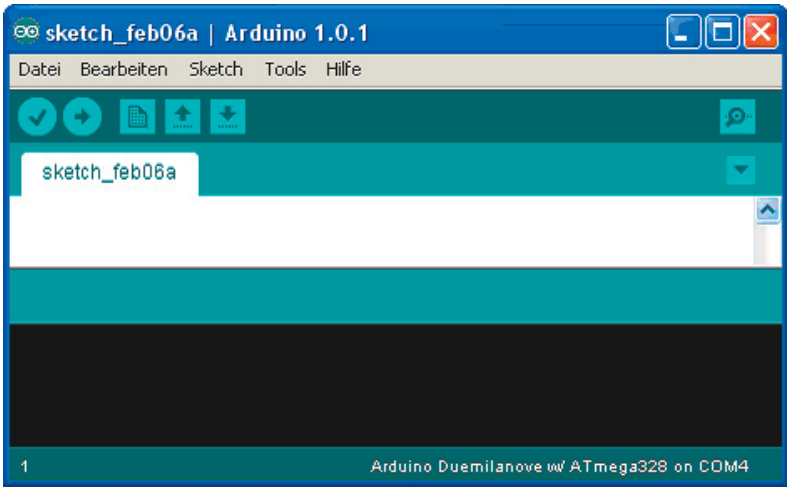

Bild 6: Unten rechts werden in der Arduino-IDE die Art des gewählten Arduino-Boards und der zugeordnete Port angezeigt.

Im Verzeichnis

C:\arduino-1.0.1\hardware\tools\avr\bin

sollte eine Sicherungskopie von *avrdude.exe* erstellt werden.

Dann wird die Datei *avrdude.exe* ersetzt durch die Version aus avrdude-5.11svn-20111019.zip (siehe [1]). Der Name muss auf jeden Fall auch wieder *avrdude.exe* lauten.

Achtung: Die Datei *avrdude.conf* nicht ersetzen!

Wenn die originale *avrdude.exe* verwendet wird, dann dauert der Programmiervorgang sehr lange, da dort die kompletten 32 KBytes des Flashs programmiert werden und nicht nur der wirklich benutzte Bootloader-Bereich.

Im Verzeichnis

C:\arduino-1.0.1\hardware\arduino

werden dann in der Datei *programmers.txt* folgende drei Zeilen hinzugefügt:

elvuino.name=ELVuino
elvuino.communication=serial
elvuino.protocol=elvuino

Nun kann der Bootloader geflasht werden. Hierzu muss die bereits besprochene Beschaltung des ATmega328P mit dem UM2102 entsprechend des Schaltplans zur Programmierung des Bootloaders (Bild 3) vorgenommen werden.

Die Arduino-Standard-LED 13 kann im Normalfall während der Programmierung des Bootloaders mit dem Pin SCK verbunden sein. Durch Flackern der LED wird dann der Programmiervorgang optisch angezeigt. Falls es jedoch zu Problemen kommt, sollte die LED während der Programmierung entfernt werden.

In der Arduino-IDE wählt man unter „Tools“ nun Folgendes aus:

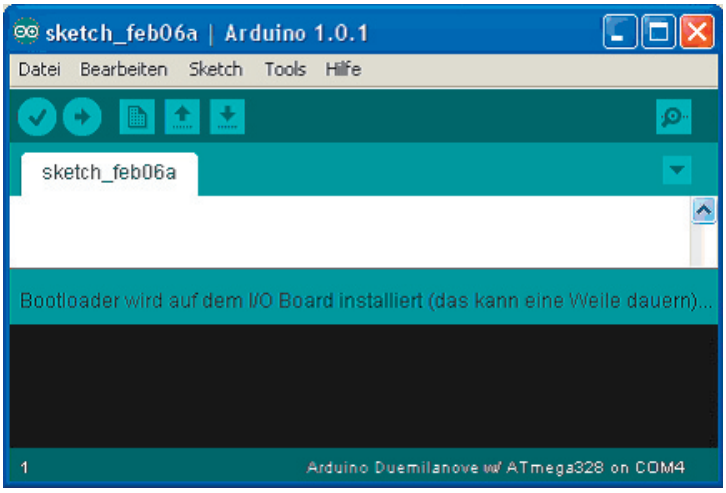

Bild 7: Die Anzeige in der IDE während der Bootloader-Installation

C:\arduino-1.0.1\hardware/tools/avr/bin/avrdude -CC:\arduino-1.0.1\hardware/tools/avr/etc/avrdude.conf -v -v -v -v -patmega328p -celvuino -P\\.\COM4 -Uflash:w:C:\arduino-1.0.1\hardware\arduino\bootloaders\atmega\ATmegaBOOT_168_atmega328.hex:i -Ulock:w:0x0F:m –i100

Bild 8: Über diese Kommandozeile erfolgt der Aufruf für AVRDUDE, um über den an z. B. COM4 angeschlossenen ELVuino den Flash eines ATmega328P mit dem Arduino-Bootloader zu programmieren.

Board: Arduino Duemilanove w/ATmega328
Serieller Port: COMx passend zum UM2102
Programmer: ELVuino

Das gewählte Board und der COM-Port werden auch unten rechts in der IDE angezeigt (siehe Bild 6).

Anschließend wird über „Tools => Bootloader installieren“ der Programmiervorgang gestartet. Leider erscheint in der Arduino-IDE kein Fortschrittsbalken, sondern nur ein Text „Bootloader wird auf dem I/O Board installiert“ (Bild 7).

Die Programmierung dauert relativ lange: ca. 1 min für die Programmierung und noch einmal ca. 1 min für den Verify-Vorgang. Dies liegt daran, dass die RS232-Handshake-Signale stimuliert werden müssen und aufgrund des USB-Protokolls nur ca. jede Millisekunde ein Wechsel der Signalzustände erfolgen kann. Die Programmierung erfolgt per „Bitbanging“, da der AVR im Auslieferzustand ja sonst per ISP programmiert werden muss.

Da der Bootloader auf jedem Controllerchip nur einmalig programmiert werden muss, spielt es aber keine große Rolle, wenn der Programmiervorgang etwas länger dauert.

Falls es zu Problemen mit der Bootloader-Programmierung über die Arduino-IDE kommt, kann auch ein manueller Aufruf über die Kommandozeile wie in Bild 8 aufgeführt erfolgen. Die Angabe „–i100“ am Ende bedeutet, dass ein Delay von 100 µs eingefügt wird. Bei manchen PCs kann es ohne diese Angabe zu einer Fehlermeldung beim Verify kommen, obwohl der Controller korrekt programmiert wurde. In der Arduino-IDE kann dieser Delay-Wert leider nicht aktiviert werden.

Nach erfolgreicher Programmierung des Bootloaders erscheint die Meldung „Bootloader wurde installiert“ (Bild 9).

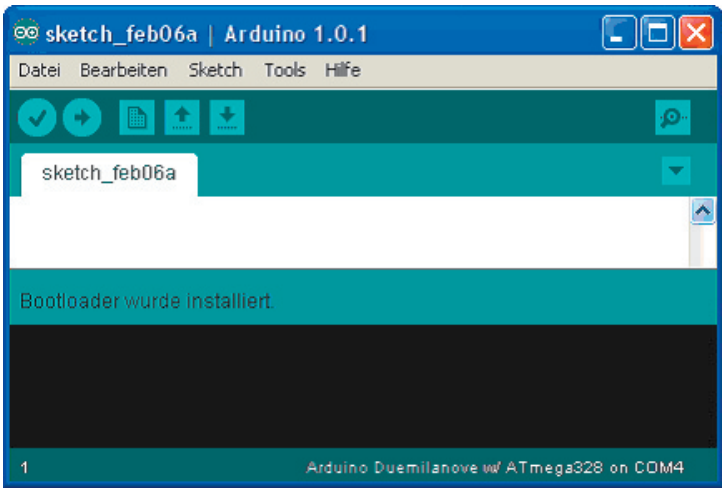

Bild 9: Hat alles funktioniert, meldet die IDE eine erfolgreiche Bootloader-Installation.

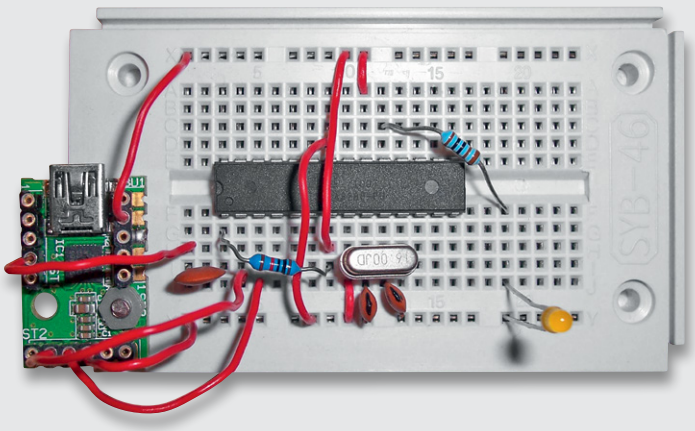

Bild 10: Die nach Bild 1 aufgebaute Schaltung zur Übertragung von Sketches auf den mit dem Bootloader versehenen ATmega328

Die Arduino-LED 13 am Pin 19 des ATmega328P sollte nun zyklisch für ca. 100 ms kurz blitzen, gefolgt von einer etwas längeren Pause von ca. 1,3 s. Dies erfolgt durch den Bootloader, um zu signalisieren, dass keine Firmware (Sketch) im Flash gefunden wurde.

Hinweise zum Aufbau der Programmierschaltung

Wenn das Laden des Bootloaders korrekt funktioniert hat, werden die Verbindungen vom UM2102 zu SCK, MOSI, MISO und Reset entfernt. Stattdessen stellt man nun die Verbindungen gemäß dem Schaltplan in Bild 1 her. Bild 10 zeigt den praktischen Aufbau.

Die Verbindung von DTR zum Reset-Pin erfolgt über einen 100-nF-Kondensator. Hierüber wird vor dem Übertragen eines Arduino-Sketches ein Reset-Puls für den AVR ausgelöst.

Der ELVuino verhält sich nun wie ein Arduino Duemilanove mit ATmega328P.

Ein Sketch kann wie bei Arduino üblich über „Upload“ von der Arduino-IDE auf den Atmega328P übertragen und gestartet werden.

Falls man Shields anschließen möchte, kann das Buchsenleisten-Set für Arduino-Boards (siehe Stückliste) verwendet und über angelötete isolierte Schaltdrähte mit dem Steckbrett verbunden werden.

Die Versorgungsspannung von 5 V wird am UM2102 direkt vom USB-Port abgegriffen.

Zum Experimentieren empfiehlt es sich, den ELVuino über einen USB-Hub an den PC anzuschließen, da dann im Falle eines Schaltungsfehlers normalerweise der PC keinen Schaden nimmt, sondern nur der Hub.

Eine 3,3-V-Spannungsversorgung wird am UM2102 ebenfalls bereitgestellt. Falls diese verwendet wird, muss darauf geachtet werden, dass nicht mehr als 100 mA Strom entnommen werden dürfen.

So vorbereitet kann der ELVuino nun über die Arduino-IDE normal programmiert werden.

Bild 11 zeigt einen Beispielsketch, der die LED 1 (D13) zum Blinken bringt.

```
int led = 13;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(250);
  digitalWrite(led, LOW);
  delay(250);
}
```

Bild 11: Ein Beispielsketch, der die LED 1 (D13) zum Blinken bringt

Weitere Infos:

[1] AVRDUDE-Version für Windows:
www.mikrocontroller.net/attachment/123832/avrdude-5.11svn-20111019.zip

[2] Infos zu AVRDUDE:
www.mikrocontroller.net/articles/AVRDUDE

[3] Arduino IDE: arduino.cc/en/Main/Software

[4] UM2102: www.elv.de/mini-usb-modul-um2102-komplettbausatz.html

Empfohlene Produkte/Bauteile:	Best.-Nr.	Preis
Mini-USB-Modul UM2102, Komplettbausatz	JX-09 18 59	€ 5,95
ATmega328P	JX-10 77 37	€ 3,–
Quarz, 16 MHz	JX-10 13 11	€ 0,18
2x Kondensator, 22 pF	JX-00 25 33	€ 0,04
Kondensator, 100 nF	JX-00 18 46	€ 0,06
Widerstand, 10 kΩ	JX-00 63 13	€ 0,45
Widerstand, 1 kΩ	JX-00 63 32	€ 0,45
Widerstand, 100 Ω	JX-00 63 11	€ 0,45
LED, gelb, 3 mm	JX-00 62 53	€ 0,10
Präzisions-Sockelstreifen, 1 x 20-polig, zum Einlöten in das UM2102	JX-06 93 21	€ 1,95
Steckboard (inkl. Drahtbrücken)	JX-05 82 46	€ 12,95
Buchsenleisten-Set für Arduino-Boards	JX-10 36 21	€ 0,75

Alle Infos zu den Produkten/Bauteilen finden Sie im Web-Shop.
Preisstellung Juni 2013 – aktuelle Preise im Web-Shop