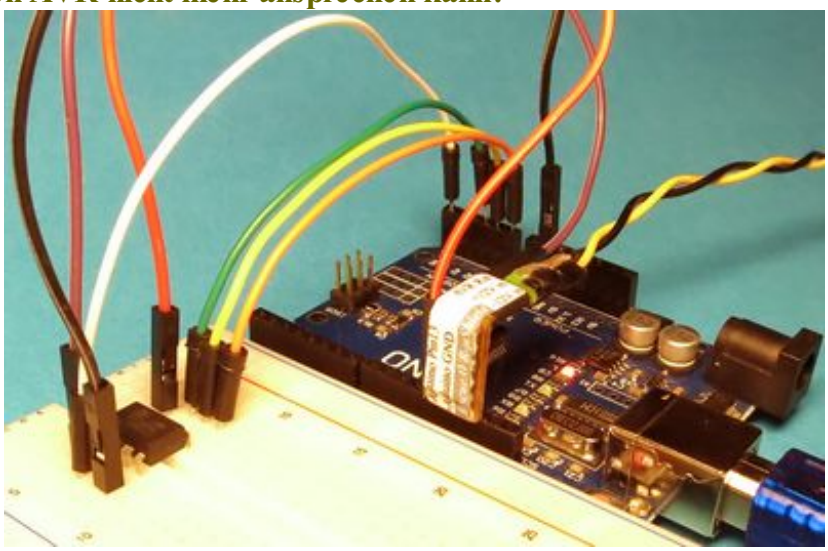
RescueAVR-MikroShield - ein Arduino als Fusebit-Resetter

Was tun, wenn unser Programmer den AVR nicht mehr ansprechen kann?



Es gibt ja schon etliche Geräte, die verfuste AVRs wieder zur Zusammenarbeit mit unserem Programmer überzeugen können. Neben einigen Programmern, die High-Volt-Programming beherrschen (nicht meine Preisliga) gibt es z.B. den Fusebit-Doctor. Bei den heutigen Preisen für Mikrocontroller müssen wir uns allerdings fragen, ob es den Aufwand überhaupt lohnt. Schön wäre, wenn es minimalistischer ginge. Wenn wir uns nicht ganz blöd anstellen, brauchen wir sowas ja auch nur selten.

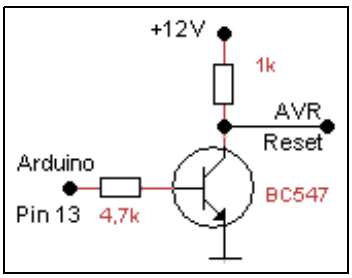
So ein Arduino hat doch eigentlich schon fast alles, was es braucht, außer die 12V Spannungsversorgung. Hat da schon mal jemand was konstruiert? Ja, es hat!

- Das HV Rescue Shield und HV Rescue Shield 2
- RescueAVR auf fritzing.org und Github

Der RescueAVR war ziemlich genau das, was ich mir vorgestellt hatte. In der Projektbeschreibung ist dargestellt, dass die komplette Elektronik auf einem Steckbrett aufgebaut ist. Die Schaltung lässt sich in zwei Teile unterteilen: Erstens einen Haufen Verbindungen (bis zu 16 bei HV-Parallelprogramming) zum AVR. Zweitens der Steuerung der 12V auf den RESET-Eingang des zu heilenden AVRs (deswegen heißt es High-Volt-Programming). Dieser zweite Teil besteht in dieser Version lediglich aus zwei Widerständen und einem NPN Transistor; er ist für alle Anwendungen immer gleich. Der Vorgänger ("RescueAVR-MiniShield") bestand noch aus 5 Bauteilen. Wir können alles auf einem Steckbrett aufbauen oder die drei Bauteile plus Anschlüsse auf eine Lochrasterplatine löten. Diese sollte das passende Format zum Aufstecken auf die Buchsenleiste von Standard Arduinos haben. Die 12V müssen wir aus einer externen Quelle zuführen (Netzteil, Batterie, o.ä.).

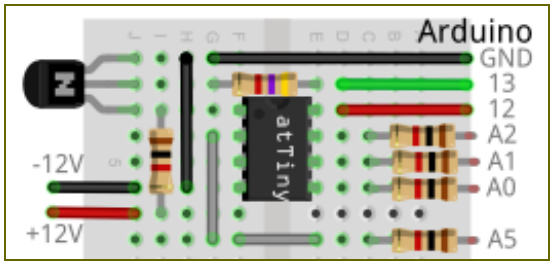
Im Bild sehen wir, wie mein 2,70 Euro China-Arduino-Uno einen ATtiny85 bearbeitet, bei dem ich den RESET-Pin deaktiviert habe,um ihn als normalen I/O-Pin zu benutzen. Die Programmierung war durch einen vorher(!) installierten Bootloader möglich, aber nicht mehr per ISP. Mittels RescueAVR können wir das nun jederzeit wieder rückgängig machen. Der UNO wird über USB mit Spannung versorgt. Die 12V kommen über das schwarz/orange Kabel auf das "Shield".

Die Schaltung



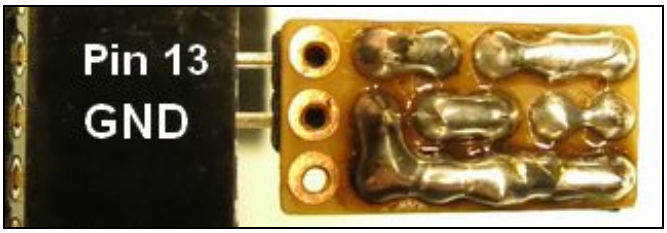
Die High-Volt Steuerung des RescueAVR habe ich abgewandelt, um sie auf ein absolutes Minimum reduzieren zu können. Der entsprechende Schaltungsteil ist links abgebildet. Die Funktion ist ganz einfach: Über den 1kOhm-Widerstand liegen 12V am RESET-Pin unseres AVR. Wird HIGH-Pegel auf Pin 13 des Arduino gegeben, werden die 12V auf des RESET-Pin unseres AVR durch den Transistor auf Masse gezogen.
Diese hardwaremäßige Vereinfachung hat zur Folge, dass sich die Ansteuerungslogik gegenüber dem originalen RescueAVR invertiert ist. Wir müssen deshalb für diese Version ein angepasstes Programm ("Sketch") benutzen. Den Download gibt es unten auf dieser Seite.

Direkt auf dem Steckbrett



Bei der geringen Anzahl erforderlicher Bauteile bietet sich an, ein Steckbrett ("Breadboard") zu verwenden. So wie hier dargestellt, könnte die Beschaltung eines ATtiny 45/85 aussehen. Die vier Widerstände an der rechten Seite sind für die Funktion nicht erforderlich. Sie schützen unseren Arduino bei Verdrahtungsfehlern oder defekten Controllern. Die 12V müssen wir aus einer separaten Quelle zuführen.

... oder ein "Shield" bauen?



Aufbau:
Ich habe natürlich wieder meine geliebten Lochrasterplatinen benutzt. Der Aufbau ist links dargestellt. Wie bei der Steckbrett-Variante müssen wir auch hier die 12V separat zuführen. Im Bild ganz oben auf der Seite ist noch ein Aufkleber zu sehen. Dieser ist auf Inkjet-Papier ausgedruckt, oben mit klarem Packband versiegelt und auf die Platine geklebt. Hauptzweck ist, die Belegung der Pins darzustellen.

Funktionstest:
Bevor wir unser Werk mit dem Arduino verbinden, machen wir natürlich einen Funktionstest. Dazu schließen wir 12V an den dafür vorgesehenen Eingang an. An "AVR Rst" kommt ein Multimeter. Außerdem halten wir eine 5V Spannungsquelle bereit. Alle drei Massen verbinden wir mit dem GND-Pin. Wenn wir jetzt +5V an den "Pin 13" Anschluss geben muss unser Multimeter 0V anzeigen. Entfernen wir die +5V muss die Spannung auf ca. 12V gehen.

Den AVR verdrahten

Unser "Shield" stellt ja lediglich den High-Volt (HV) Ausgang zur Verfügung. Der geht zum RESET Pin des AVR. Alle anderen Verbindungen zum AVR kommen direkt vom Arduino Bord; ggf. über ....
Angstwiderstände:
Keine schlechte Idee ist es, die Datenleitungen (also alles was an Verbindungen direkt vom Arduino zum AVR geht, außer *HV out*, *Vcc* und *GND*) mit Schutzwiderständen zu versehen. RescueAVR nutzt keine,

der Fusebit-Doctor 1 kOhm in jeder Leitung. Auf der sicheren Seite ist, wer welche benutzt.

Welche Verbindungen wir noch herstellen müssen, verraten uns die Bilder, die im Github Download enthalten sind oder die folgenden Tabellen. Die Tabellen decken nur die gängigsten Modelle ab. Insgesamt kennt RescueAVR 212 AVR-Modelle!

ATtiny x5		
Pin	Funktion	Arduino
1	RESET	AVR Rst
2	SCI	A5
3	-	-
4	GND	GND
5	SDI	A0
6	SII	A1
7	SDO	A2
8	Vcc	12

ATtiny x4		
Pin	Funktion	Arduino
1	Vcc	12
2	SCI	A5
3	-	-
4	RESET	AVR Rst
5	-	-
6	-	-
7	SDI	A0
8	SII	A1
9	SDO	A2
10	-	-
11	GND	GND
12	GND	GND
13	GND	GND
14	GND	GND

ATmega x8					
Pin	Funktion	Arduino	Pin	Funktion	Arduino
1	RESET	AVR Rst	15	DATA1	8
2	-	-	16	DATA2	7
3	RDY	A4	17	DATA3	6
4	OEi	A3	18	DATA4	5
5	WRi	A2	19	DATA5	4
6	BS1	A1	20	Vcc	12
7	Vcc	12	21	-	-
8	GND	GND	22	GND	GND
9	XTAL1	A0	23	DATA6	3
10	-	-	24	DATA7	2
11	XA0	10	25	BS2	A5
12	XA1	11	26	-	-
13	PAGEL	GND	27	-	-
14	DATA0	9	28	-	-

Bedienung

```
RescueAVR   Version  2.1i
interactive, running on Arduino

Signature: 1E930B
MCU name:  ATtiny85
Current L/H/E-Fuses: E2 7F FF
Default L/H/E-Fuses: 62 DF FF
Current lock byte:   FF
Oscillator calibr.:  56

Choose:
  R - Try to resurrect chip by all means
  E - Erase chip
  D - Burn default fuse values
  L - Change low fuse
  H - Change high fuse
  X - Change extended fuse
  K - Change lock byte
Action:
```

RescueAVR kommt völlig ohne Hardware-Bedienelemente aus. Die gesamte Kommunikation wird über UART und den sowieso im Arduino verbauten USB-seriell Schnittstellwandler abgewickelt. Das heißt, wir müssen den Arduino per USB mit einem Computer verbinden und computerseitig ein Terminal-Programm am Start haben. Ist bei uns die Arduino IDE installiert, haben wir schon alles an Bord (dort wählen wir *Werkzeuge - Serieller Monitor*).

Die Schnittstellenparameter lauten: 19.200 Baud, 8 Datenbits, keine Parität, 1 Stoppbit.

Als Nummer der seriellen Schnittstelle wählen wir die unseres Arduino Bords aus. Notfalls müssen wir im Windows-Gerätemanager nachsehen. Bei den ganz billigen China-Klonen (wie bei mir) ist in der Regel ein CH340 Schnittstellenwandler verbaut. Der ist bei mir als "*USB-SERIAL CH340 (COM5)*" angemeldet.

Haben wir alles fehlerfrei aufgebaut und angeschlossen, sollten wir in unserem Terminalprogramm eine Meldung sehen wie im Bild sehen. Über die Sendefunktion unseres Terminalprogramms geben wir die entsprechenden Anweisungen und erhalten dann eine neue Meldung.

Software & Downloads

Download Software Version 2.1i (enthält Quelltext, vorkompilierte hexFiles, Aufkleber, Doku)

RescueAVR auf Github (ACHTUNG: Der beiliegende Quelltext funktioniert NICHT mit der hier beschriebenen Hardware):
https://github.com/felias-fogg/RescueAVR
Direkter Download:
https://github.com/felias-fogg/RescueAVR/archive/master.zip

