Joystick- Controller mit Bascom-SWUSB

Ralf Beesner

21. September 2012

1 Überblick

Ich habe noch einen alten analogen Joystick herumliegen - aber keinen Rechner mehr, der einen passenden Gameport- Anschluss aufweist. Zu neuem Leben ließ er sich mit einem AVR- Mikrocontroller und der SWUSB-Library erwecken.

Die von Rick Richards entwickelte SWUSB-Library ist das Bascom-Gegenstück zur VUSB-Library von `http://www.obdev.at/vusb`.

Der Kern, die Library swusb.lbx und ein include-File sowie (derzeit) zwei Beispielimplementationen (ein Gamepad und ein Keyboard) finden sich unter `http://www.sloservers.com/swusb`.

Ich habe sein Gamepad-Beispiel auf einen AtTiny44 bzw. AtTiny84 portiert (der Joystick hat nur 2 Achsen und 4 Taster, so dass die verfügbaren Pins eines AtTiny44 gerade so reichen).

Da das gut klappte, habe ich mit den HID-Descriptoren herumgespielt und Lösungen für umfangreichere Joysticks gefunden; z.B für Joysticks mit einem zusätzlichen Schubregler und einem HatSwitch (auch als Coolie Switch oder POV bekannt).

Eine weitere Lösung bietet 4 Achsen und 8 Taster - sie wäre geeignet, um aus der Mechanik eines abgewrackten Fernsteuersenders mit zwei Steuerknüppeln einen PC-Trainer für PC-Modellflug-Simulatoren zu bauen.

2 Hardware

Die USB-Anschaltung an den Mikrocontroller ist die gleiche wie bei den VUSB-Projekten.

Beim Attiny44/84 gibt es eine kleine Besonderheit: D+ muss an INT0 (PB2) liegen, aber auf PortB ist kein weiterer Pin frei. Glücklicherweise kann man als USB-Datenpins z.B. PA0 and PA1 nutzen und PB2 zusätzlich mit D+ verbinden. PA0 und PA1 übertragen dann die Daten, während PB2 nur die Interrupts auslöst.

Die Potis bilden einen Spannungsteiler zwischen VCC und Masse, und die Taster bzw. Schalter tasten gegen Masse.

Bei Verwendung eines alten analogen Joysticks ist jedoch zu beachten, dass die intern verbauten Potentiometer nicht als Spannungsteiler beschaltet sind, sondern lediglich einseitig an Plus 5V liegen.

Es gibt zwei Lösungsmöglichkeiten: Wenn man im Inneren des Joysticks eine Masseleitung findet, kann man den freien Anschluss der Potis an Masse legen (dabei sollte man zur Sicherheit kontrollieren, ob Plus wirklich am anderen Ende des Potis liegt und nicht etwa am Schleifer des Potis!). Ist dies nicht der Fall, kann man auf der Platine die ADC-Eingänge mit etwa 10 kOhm gegen Masse ziehen. Der Joystick-Widerstand von 100 kOhm und der 10-kOhm-Widerstand bilden dann einen Spannungsteiler. Nachteilig ist, dass die Spannung dann nicht mehr linear vom Drehwinkel des Joystick-Potentiometers abhängt und minimal 0,5V statt 0V beträgt.

Die Joystick-Potis überstreichen ohnehin nicht den gesamten Widerstandsbereich, sondern nur einen Teil (daher sehen die Joystick-Treiber bzw. die Anwendungsprogramme eine Joystick-Kalibrierung vor, so dass man den verschobenen Nullpunkt "wegkalibrieren" kann).

Der AtMega8-Schaltplan stellt den "Hardware-Maximalausbau" für 4 Achsen

und 8 Taster / Schalter dar.

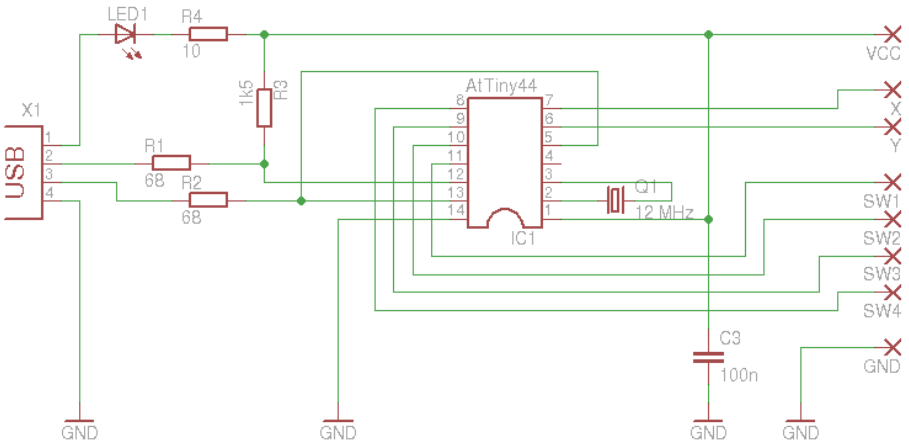

Abbildung 1: Schaltbild Zweiachs- Vierschalter-Version

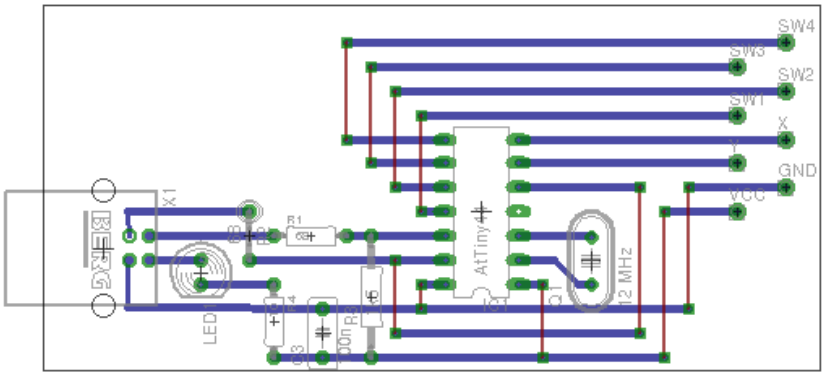

Abbildung 2: Streifenraster- Layout

3 Software

Ollopas Gamepad-Beispiel-Implementation habe ich nur ansatzweise verstanden, aber glücklicherweise erkennt man leicht die Stellen, an denen man sich in Ollopas Code mit seiner eigenen Task einhängen kann.

Das größte Hindernis war auch hier, an funktionierende HID-Deskriptoren für die verschiedenen Joystick-Varianten zu kommen. Linux ist recht tolerant, aber Windows bemeckert jeden kleinen Fehler mit "der Treiber kann nicht gestartet werden (Code 10)", obwohl die HID-Spezifikation eigentlich vorsieht, dass das Betriebssystem unverständliche Teile übergehen und den Rest ausführen soll.

Eine gewisse Hilfe war ein HID-Parser namens `Dt.exe` bzw. `dt_2.4.zip` , der unter `http://www.usb.org/developers/hidpage` zu finden ist. Die syntakti-

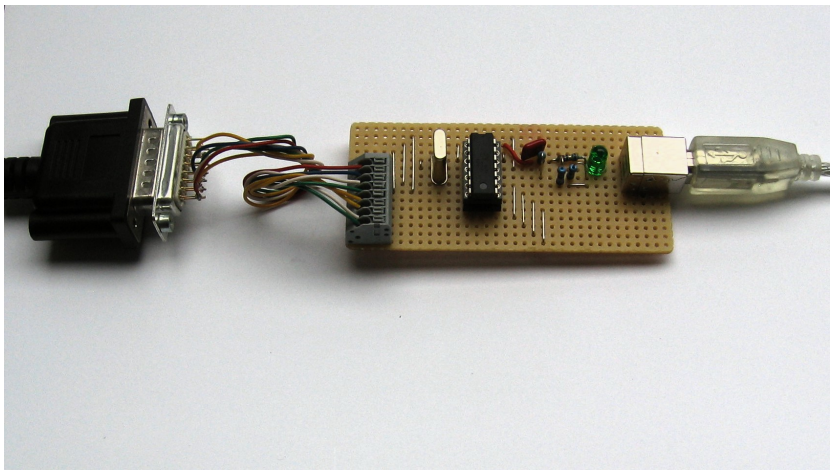

Abbildung 3: Platine

sche Prüfung durch das Tool ist zwar eher nutzlos, aber man kann sich einige Beispiele für funktionierende HID-Deskriptoren herauskopieren.

Die Länge eines geänderten HID-Deskriptors ist im Bascom-Code an zwei (!) Stellen anzupassen (die zweite ist knapp 20 Zeilen oberhalb des eigentlichen Deskriptors).

Den zentralen Teil, die Abfrage der AD- Wandler und das Umsetzen in USB-Telegramme, findet man um Zeile 270 herum; bei Weglassen oder Hinzufügen einer Achse muss ein Byte weniger (bzw. mehr) in das Array `_usb_tx_buffer2` geschrieben werden. Außerdem muss die Anzahl der Nutzbytes angepasst werden, die beim Aufruf der Funktion `usb_send` übergeben wird (also die Zahl hinter `_usb_tx_status2`).

4 Kompilieren des Codes

Wegen der restriktiven Lizenz für swusb.lbx habe ich dem Sourcecode lediglich das File swusb-includes.bas beigefügt - swusb.lbx ist auf der oben angegebenen Homepage des Projekts herunterzuladen und in den BASCOM- Plugin-Ordner zu verschieben.

5 Fusen des AtTiny

Ein fabrikfrischer Attiny muss für Quarzbetrieb ohne 1:8- Vorteiler gefust werden; Watchdog und Browout sind nicht aktiviert:
Low Fuse = 0xFF High Fuse = 0xDF

Für AtMega8 lauten die Fusebytes:
Low Fuse = 0xFF High Fuse = 0xD9

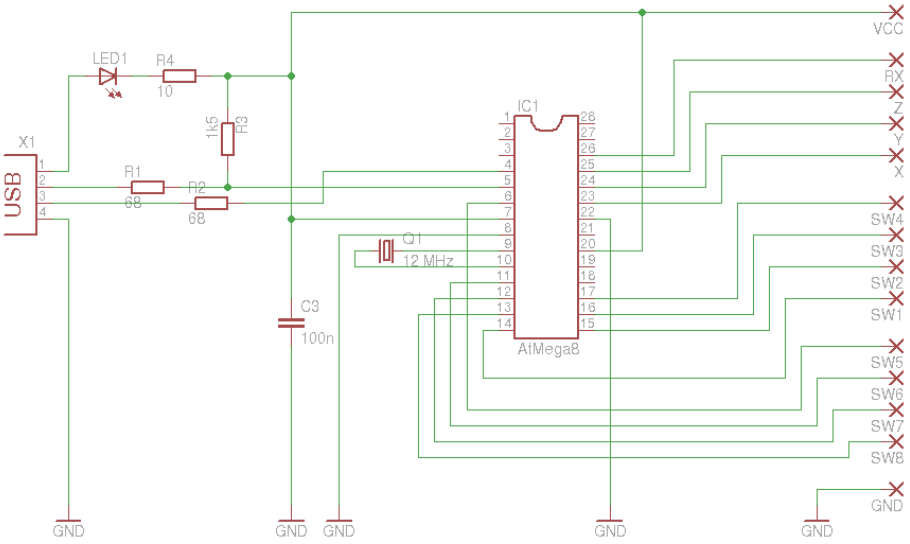

Abbildung 4: Schaltbild Vierachs- Achtschalter-Version

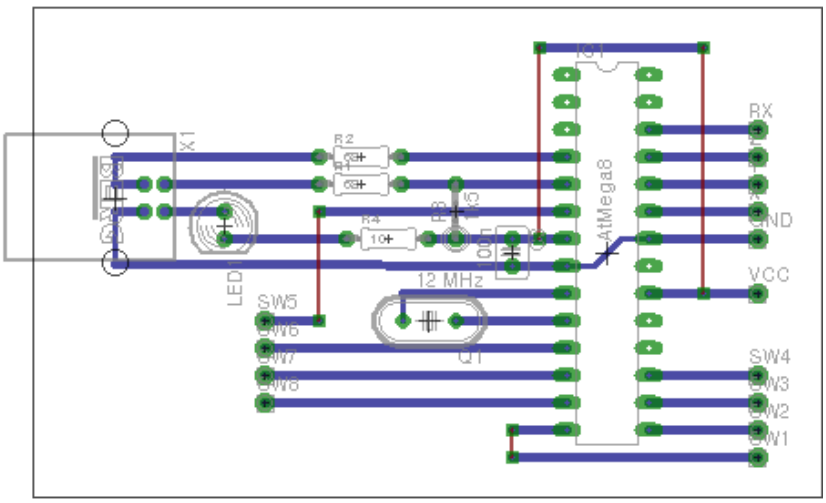

Abbildung 5: Streifenraster- Layout