uC Einführungsworkshop

Es soll ein Workshop bzw. eine kleine Workshopserie als Einführung in die Programmierung von AVR Mikrocontrollern ausgearbeitet werden.

Projektkurzinfos

contact
: alex

begin
: 2013-01-03

status
: planning/preparing

skillz (source)
: Vorbereitung erfordert für den Lehrenden Ahnung von Schaltung aufbauen, AVR programmieren in C, Makefiles schreiben, Teileilisten machen, Bausätze vorbereiten, Folien ausarbeiten, … Workshop durchführen needs Erklärbärfähigkeit, Geduld, …

skillz (drain)
: Der Lernende muss als Voraussetzung mitbringen: sicherer Umgang mit Editor und Kommandozeile, Programme nachinstallieren (avrdude, avr-gcc), Basiswissen Stromkreise (Physik 9. Klasse), Grundkenntnisse Computerprogramme (Variablen, Schleifen, Funktionen)

Ideen

Für die Einführung ist der ATtiny 25/45/85 ganz gut geeignet, übersichtliche Anzahl an Funktionseinheiten, billig zu beschaffen und als DIP-8 oder SOIC-8 erhältlich. Eine einfache Beispielbeschaltung passt locker auf ein kleines Whiteboard:

- [tux, 2012-04-03] Wie wir heute festgestellt haben, ist der ATTINY 24A-PU eine durchaus brauchbare Alternative mit mehr Ports zu einem vernachlässigbar höheren Preis. Entsprechend der untenstehenden WS-Gliederung könnten wir überlegen, diesen µC zu verwenden und mit einer RGB-LED sowie anderer Zusatzbeschaltung (drei Potis, damit man mit RGB und Farbräumen herumspielen kann, …) zu versehen.

Im Laufe der Vorbereitungen stellte sich heraus, dass man die Thematik in mehrere Workshops gliedern sollte. Einführung in die Mikrocontrollerprogrammierung kann mit nahezu gleichem Quellcode auf Steckbrett, gelöteter Schaltung (Lochraster, SMD) oder Arduino geschehen. Ansteuern kann man diverse Dinge, der Einfachheit halber bieten sich LEDs als Ausgabe und Potis als Eingabe an.

Den Aufbau einer der Schaltungsvarianten kann man in je einen Extra-Workshop packen. Der Umfang ist mit LED und Poti mehr als ausreichend für einen Einführungsworkshop.

Alternative Beispielschaltung mit 7-Segment-Anzeige



Hinweis: Wegen $grund ist an PB4 (Pin 3) der Pull-Up-Widerstand nicht wirksam. Deshalb wurde ein externer Widerstand (10kOhm) von Pin 3 nach Masse gelegt.

Der AT90S2343 wurde gewählt, weil er im Bastelbestand noch vorhanden war. Letztendlich sollte auch hier ein ATtiny25 zum Einsatz kommen. Folgende Varianten sind dann möglich:

- Der 4bit-Hex-Codierer wird durch ein Poti ersetzt. Der Stellwert wird am ADC-Eingang ermittelt.
- Einer der frei werdenden Ports kann dann wieder für den Reset des Zählers verwendet werden (d.h., wieder zwei Leitungen zum Zähler, aber eine robustere Schaltung). Diese beiden Schaltungsvarianten mit Vor- und Nachteilen können durchaus auch im Rahmen eines Workshops behandelt werden.
- Das Poti lässt sich auch durch einen widerstandsbasierten Sensor ersetzen, z.B. einen Photowiderstand. Auf der Anzeige erscheint dann ein Lichtlevel.
- Mit einer höheren Taktung lässt sich das Flackern der 7-Segment-Anzeige wahrscheinlich völlig unterbinden. (Konsequenzen verschiedener Taktungen können ebenfalls Teil eines Workshops sein.)

Bauteile

Im Prinzip sind die Teile sehr ähnlich, egal welche Schaltung man benutzt. Leichte Abweichungen vielleicht beim Arduino. Interessant für die gelöteten Schaltungen wäre vielleicht noch ein vorgeschalteter Spannungsregler.

Schaltung auf Steckbrett

- ATtiny25/45/85 in DIP-8 Gehäuse
- ein bis drei LED oder eine RGB-LED (bspw. LED RGB-5 DIFFUS)
- ein paar Widerstände und Kondensatoren
- ein Poti
- ein Adapter zum Anstecken des Programmers
- kleine low-current LED als Betriebsspannungsanzeige

Schaltung auf Lochraster

Kann man die selben Teile nehmen wie auf dem Steckbrett.

SMD-Schaltung

Setzt das Projekt PCB-Herstellung voraus, kann von der Schaltung her aber ähnliches bis gleiches Design nutzen. Zur Demonstration nimmt man dann einfach den Mikrocontroller im SOIC-8 Gehäuse.

Lerneinheiten

In zwei Stunden lässt sich das nicht abfeiern, daher Unterteilung in mehrere Blöcke. Möglich wären …

1. alles bis »Hello World« wie bspw. Steckbrett aufbauen, Platine löten, Arduino auspacken als parallel mögliche Einheiten
    1. Steckbrett aufbauen
    2. Schaltung löten
    3. Grundlagen Stromkreise
2. »Hello World« aka LED blinken lassen mit vorbereitetem Makefile und uC Flashen durch die Teilnehmer, der Teil wurde bereits für die OHM2013 vorbereitet und durchgeführt
    - Struktur eines C-Programms
    - Zugriff auf Ports des uC
    - Datenblatt lesen in Kürze
    - kompilieren und flashen
    - ggf. Einbindung der `delay.h`
3. LED dimmen mit PWM
    - kurze Theorie zu LED dimmen, Strom, Spannung, PWM
    - Vorstellung Timerbaustein des uC
    - Interrupts und ISR
    - einfache PWM mit Timer und ISR
    - Lookuptable für schickes Dimmen
4. Einlesen von Daten von außen
    - Grundlagen AD-Wandler
    - genauere Beschäftigung mit uC-Registern
    - Poti anschließen, einlesen und dann die Helligkeit der LED steuern
5. Combine all previous stuff with RGB!
    - HSV
    - was tun bei common anode?
    - Optimierung der PWM

Ressourcen/Links

- Datenblatt: http://www.atmel.com/Images/doc2586.pdf
- Application Notes
    - AVR125: ADC of tinyAVR in Single Ended Mode
    - AVR130: Using the timers on tinyAVR and megaAVR devices
    - AVR131: Using the 8-bit AVR High-speed PWM
    - AVR136: Low-jitter Multi-channel Software PWM
- Pad
- altes Pad
- Repository
- Follow-Up: AVR 102
- notizen_avr101.pdf

From:
https://wiki.netz39.de/ - **Netz39**

Permanent link:
https://wiki.netz39.de/projects:2013:avr_101

Last update: **2013-08-06 12:40**