pentabug

beep, rattl, narf

2bt, bigalex, _john

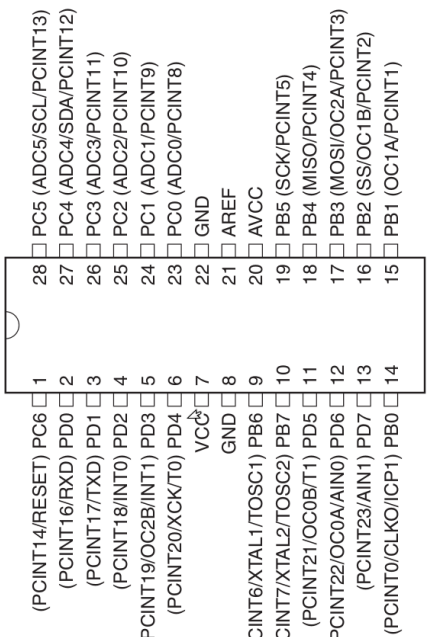
Oktober 2012

What the Bug!

- Because we can and thought it's fun.
- Interessantes und günstiges Projekt für den Junghackertrack.
- Nutzung für CMS.
- Einfacher, spielerischer Einstieg in uC-Technik (Hard- und Software).
- Soll Spaß machen und viele Altersstufen ansprechen.
- Ausbaufähig und nachhaltig.
- Viele verschiedene, teilweise komplexe Herausforderungen.

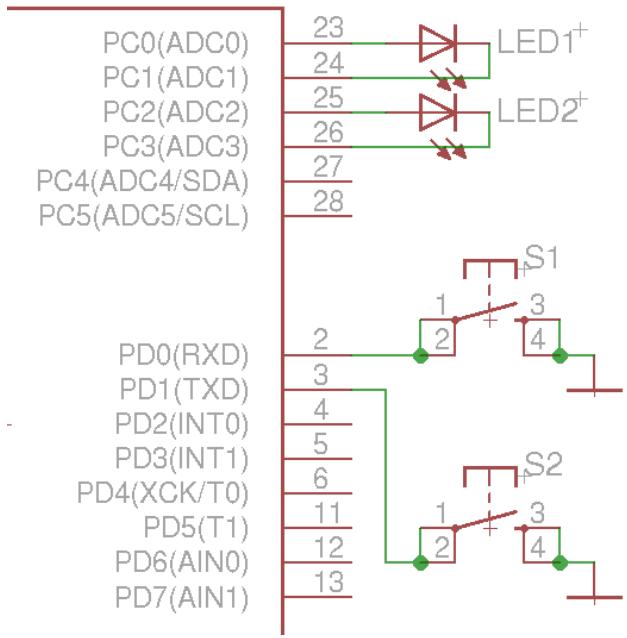
Prozessor: ATMEL ATmega88P

- Preiswert (1,20 EUR), weit verbreitet, einfach zu programmieren.
- Viele Beispiele und Tutorials im Netz.
- Nur wenige Peripheriekomponenten notwendig.
- PDIP 28 - Gehäuse, leicht zu löten.
- 8-bit AVR RISC
- 8KB flash, 512B EEPROM, 1KB SRAM
- 23 GPIO, 3 CTC, USART
- 6 A/D-Wandler Kanäle (10-bit)
- 2.7-5.5 V, power save



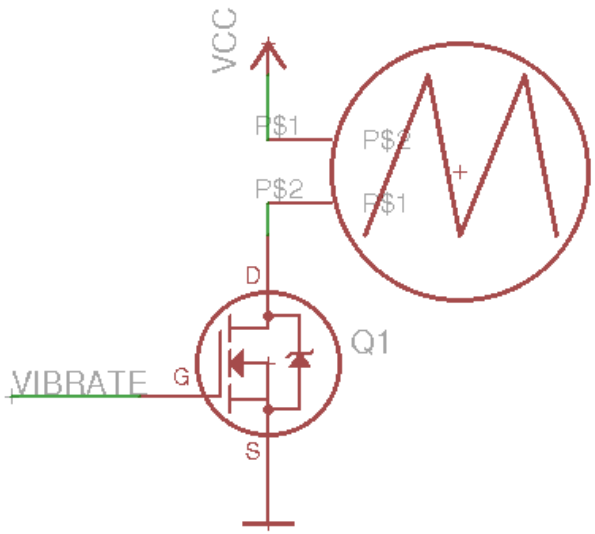
LED und Schalter

- Low Current LEDs
- Kein Vorwiderstand notwendig.
- An ADC-Pins für Helligkeitsmessung.
- Taster gegen GND (nutzen interne Pull-Ups).

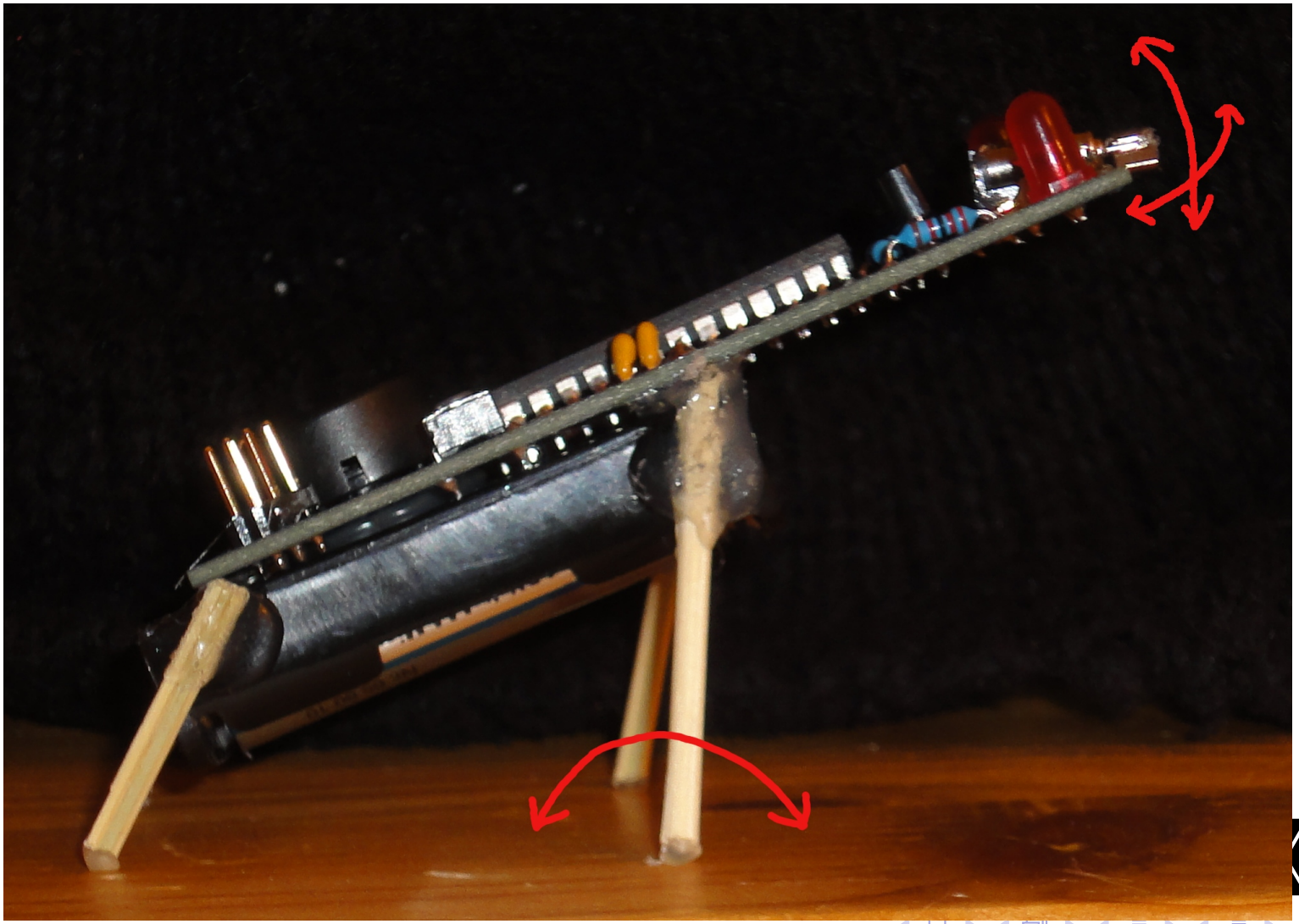


Motor

- Vibrationsmotor für Mobiltelefone.
- 2N7000 N-Kanal FET
- Kein Vorwiderstand notwendig.
- Bis 400mA
- Interne Freilaufdiode.
- Robust (ESD!) und preiswert.

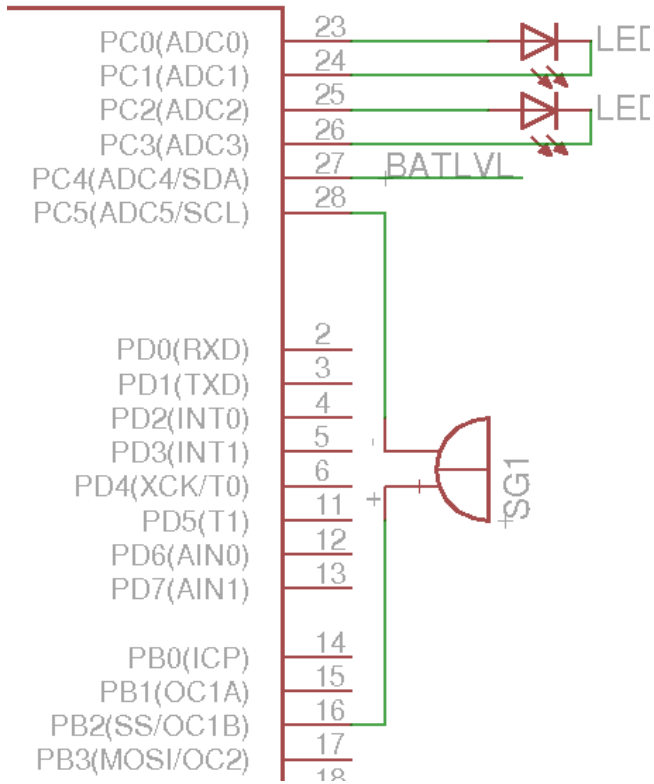


Mechanischer Aufbau



Piezo-Buzzer

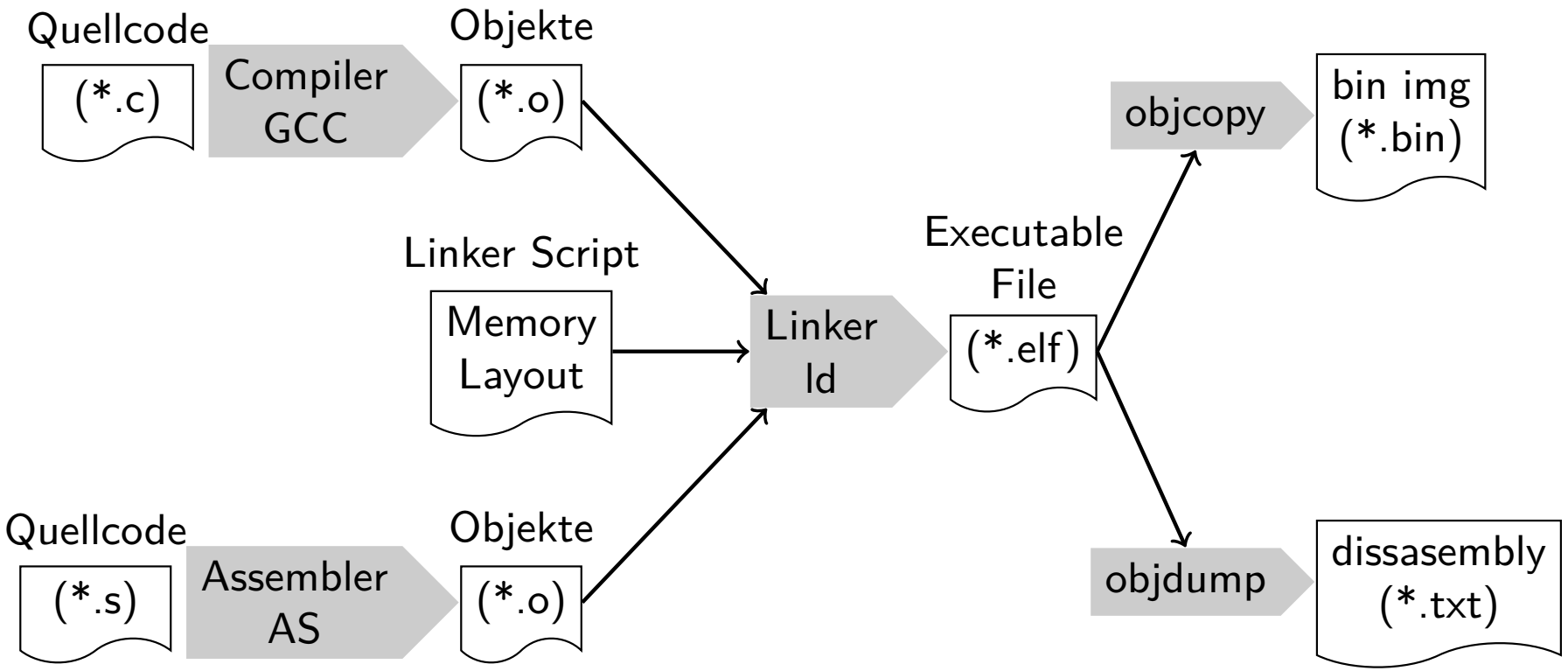
- Anschluss an zwei Pins (nicht gegen GND)
- Größerer Schallpegel
- Als Schallsensor verwendbar (ADC-PIN)
- unerwartet großer Ladungsspeicher



Sonstiges

- Programmierschnittstelle ISP
- Programmieren per USART (erfordert Bootloader)
- Bidirektionale Kommunikation per USART (Debugging, Meßwerte auslesen, ...)
- Spannungsteiler zum Messen von Vcc an ADC-Pin
- 6 Ports + Vcc + GND auf Lötpunkten für Stift-/Buchsenleisten

Programmerzeugung in C (GNU Toolchain)



Beispielcode

Minimalbeispiele

Led, Motor ansteuern:

```
1   /* Leds als Ausgang konfigurieren: */
2   DDRC |= (1 << PORTC0) | (1 << PORTC1) |
3     (1 << PORTC2) | (1 << PORTC3);
4   while (1){/* endlosschleife */
5     /* Leds ausschalten: */
6     PORTC &= ~( (1 << PORTC0) | (1 << PORTC1) |
7         (1 << PORTC2) | (1 << PORTC3) );
8     /* Sekunde warten:*/
9     _delay_ms(1000);
10    /* Leds anschalten */
11    PORTC |= (1 << PORTC0) |(1<< PORTC2);
12    _delay_ms(1000);
13  }
14 }
```

einfache Tonerzeugung

ist wie schnelles Blinken

```
1   /* Piezo als Output */
2   DDRC |= (1 << PORTC5);
3   DDRB |= (1 << PORTB2);
4   buzzr_off();
5
6   /*endlos */
7   while (1){
8     buzzr_up();
9     _delay_ms(2);
10    buzzr_inv();
11    _delay_ms(2);
12    } /* periode 4ms -> f = 1/0.004 = 250Hz */
```

einfaches Tasten abfragen

Der Pinzustand steht im PIN Register

```
1  {
2    /* Leds als Ausgang konfigurieren: */
3    DDRC |= (1 << PORTC0) | (1 << PORTC1) |
4      (1 << PORTC2) | (1 << PORTC3);
5    /* Taster als Eingang */
6    DDRD &= ~((1<<PORTD0)|(1<<PORTD1));
7    /* pullup an */
8    PORTD |= (1<<PORTD0)|(1<<PORTD1);
9    while (1){
10     /* Leds ausschalten: */
11     PORTC &= ~((1 << PORTC0) | (1 << PORTC1) |
12         (1 << PORTC2) | (1 << PORTC3) );
13     /* auf Taste warten:*/
14     while (!(PIND & 0b00000011)) { /*nichts*/ }
```

uart als debugterminal

... wenn geblinke nicht mehr ausreicht

uart als debugterminal

Analogwandlerwerte auslesen:

```
1    while(1){
2      /* neuen wert lesen: */
3      analogwert = ADCW;
4      /* wert auf serieller schnittstelle ausgeben: */
5      USART0_put_uint16(analogwert);
6      USART0_crlf(); //neue zeile...
7    }
```

Schallsensor

Der Piepslautsprecher funktioniert auch „falsch“ herum!

Led als Lichtsesor

Eine Led ist auf verschiedene Weisen als Lichtsensor zu gebrauchen:

- Sperrschichtkapazität aufladen, Fotostrom entlädt
- LED als „Solarzelle“

Level 1: Löten,Aufbau,kreative Bastelei

- Do something different and do it yourself!
- Kinder (und Erwachsene) für Technik begeistern.
- Man kan Dinge selber gestalten statt Vorgefertigtes zu verwenden.
- Erfolgserlebnisse mit niedriger Einstiegshürde.

Level 2: Computer in elektronischen Geräten

- Jeder ist von ständig von vielen(!) Computern umgeben.
- Irgendjemand bestimmt, was diese tun und wie sie etwas tun.
- Man kann damit viele Dinge tun, mit denen man nicht rechnet.

Level 3: Code selbst verstehen und gestalten

- Verstehen und nachvollziehen was Programme tun.
- Software/Hardware anpassen und ändern.
- Eigene kreative Ideen praktisch umsetzen.
- Code mit anderen teilen und gemeinsam entwicklen.

Showcase

- Komplexerer Code, der zeigt, was mit wenig Hardware alles möglich ist.
  - Polyphoner Synthesizer
  - Lichttheremin
  - Schreckbug

- Pentabug als einsteigergeeignetes uC-Projekt.
- Preiswert und einfach aufzubauen.
- Schnelle Erfolge für Einsteiger.
- Anspruchsvollere “Hacks” für Fortgeschrittene.
- Erweiterbar, z. B. für Projekte nutzbar.
- Gute Resonanz bei Junghackern und Jugendlichen
- Sa: 40 Bausätze, nahezu alle fehlerfrei aufgebaut.

Danke fürs Zuhören! Fragen?


C3D2 auf github.
URL: `http://github.com/c3d2/pentabug`.


Mikrocontroller AVR-GCC-Tutorial.
URL: `http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial`.


Mikrocontroller AVR-Tutorial.
URL: `http://www.mikrocontroller.net/articles/AVR-Tutorial`.