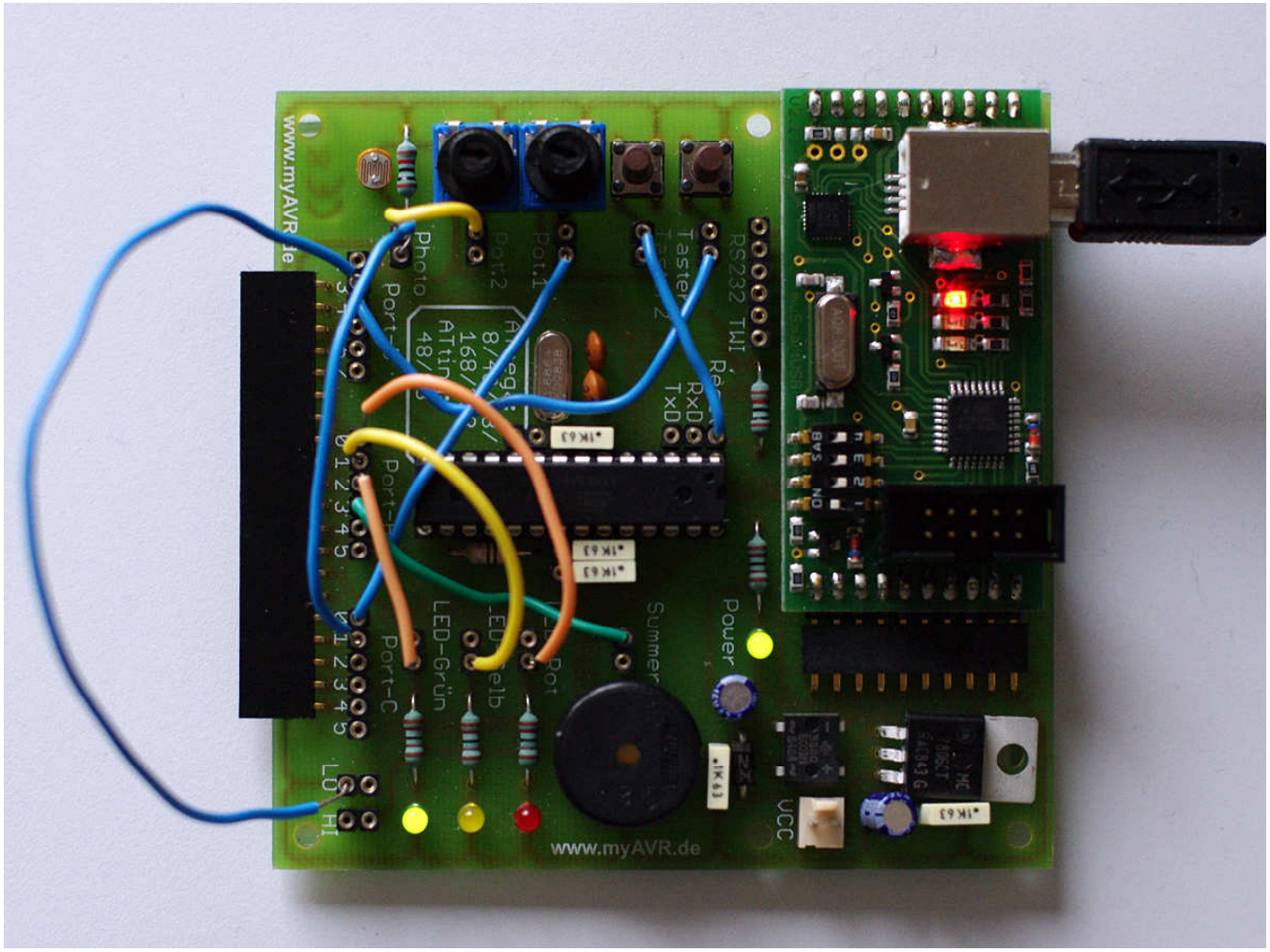
Projekt: Alarmanlage mit Sirene

Alarmanlage mit regelbarer auf und abschwellender Sirene, ausgelöst durch Lichteinfall oder Unterbrechung einer Alarmschleife. Die Lichtempfindlichkeit ist einstellbar. Die Anzahl ausgelöster Alarme wird gespeichert und durch Blinken einer LED dargestellt – auch bei Stromunterbrechung.



Quelltext

```
;+---------------------------------------------------------------------
;| Title            : myAVR Alarmanlage mit Sirene ausgelöst durch Unterbrechung einer Alarmschleife oder Licht
;+---------------------------------------------------------------------
;| Funktion         : Alarmanlage
; Sirene: Tonerzeugung mit Timer, Tonhöhe gesteuert mit OutputCompareRegister2 (OCR2)
; je kleiner der Wert in OCR2 desto schneller wird der Timer zurückgesetzt
; desto höher die Frequenz des Tons
; Die Sirene wird gestartet, wenn der Lichtsensor (LDR) beleuchtet wird, also
; Licht in einen dunklen Raum fällt
; oder wenn eine Kontaktschleife zwischen PortD3 und dem Minuspol (=Lo) unterbrochen wird
; z.B. an einen Unterbrecherkontakt

; Wenn das Gerät eingeschaltet wird, leuchtet die grüne LED
; Sobald Taster 1 gedrückt wird, erlischt LED grün und LED gelb brennt für einige Sekunden.
; Darauf blinkt die rote LED so viele Male, wie der Alarm früher aktiviert wurde.
; Anschliessend ist die Alarmanlage scharf.
; Die Sirene ertönt, sobald das Licht angeht oder der Kontaktdraht unterbrochen wird.
; Gleichzeitig mit der Sirene blinkt dir rote LED im Takt.
; Die Sirenengeschwindigkeit lässt sich mit Potentiometer 1 einstellen.
; Potentiometer 2 regelt die Lichtempfindlichkeit.
; Mit Taster 1 kann der Alarm zurückgestellt werden (wenn es wieder dunkel ist und der Kontaktdraht wieder leitet).
; Taster 2 ist der Resettaster. Der Alarmzähler wird wie folgt auf 0 zurückgesetzt:
; Taster 1 gedrückt halten und Taster 2 (Reset) kurz drücken und danach Taster 1 wieder loslassen - es leuchten
; kurz alle LEDs auf und der Alarmzähler im EEPROM wird auf 0 gesetzt.

;| Schaltung        : ...

; an Reset: Taster 2
; an PB0 : LED-Rot
; an PB1 : LED-Gelb
; an PB2 : LED-Grün
; an PB3 : Summer
;
; an PC0 : Potentiometer 1
; an PC1:  LDR (Light Dependent Resistor) UND Potentiometer 2
;
; an PD3:   Kontaktschleife (Draht) verbunden mit Lo (Masse)
; an PD4:  Taster 1
```

```
;+--------------------------------------------------------------------
;| Prozessor                          : ATmega8
;| Takt                                  : 3,6864 MHz
;| Sprache          : Assembler
;| Datum            : März 2011
;| Version          : ...
;| Autor            : Reto Savoca
;+--------------------------------------------------------------------
.nolist
.include    "m8def.inc"
.list
;--------------------------------------------------------------------
;Reset and Interrupt vector             ;VNr.  Beschreibung
            rjmp    reset               ;1   POWER ON RESET
            reti                        ;2   Int0-Interrupt
            reti                        ;3   Int1-Interrupt
            reti                        ;4   TC2 Compare Match
            reti                        ;5   TC2 Overflow
            reti                        ;6   TC1 Capture
            reti                        ;7   TC1 Compare Match A
            reti                        ;8   TC1 Compare Match B
            reti                        ;9   TC1 Overflow
            reti                        ;10  TC0 Overflow
            reti                        ;11  SPI, STC Serial Transfer Complete
            reti                        ;12  UART Rx Complete
            reti                        ;13  UART Data Register Empty
            reti                        ;14  UART Tx Complete
            reti                        ;15  ADC Conversion Complete
            reti                        ;16  EEPROM Ready
            reti                        ;17  Analog Comparator
            reti                        ;18  TWI (IC) Serial Interface
            reti                        ;19  Store Program Memory Ready
;--------------------------------------------------------------------
;Start, Power ON, Reset

reset:
     ldi     temp, HIGH(RAMEND)
     out     SPH, temp
     ldi     temp, LOW(RAMEND)          ; Stackpointer initialisieren
     out     SPL, temp
```

```
;-----------------------------------------------------------------------
;Register
.def        temp=r17                    ; Temporäre Hilfsvariable
.def        kurzverz=r18                ; Geschwindigkeit der Sirene bzw. Pausenlänge für diverse Blinker
.def        licht=r20                   ; Lichtintensität (von LDR via AD-Wandler)
.def        alarmstatus=r21             ; 0=kein Alarm  1= Alarm
.def        dataread=r22                ; Variable um Wert aus EEPROM zu lesen
.def        datawrite=r23               ; Variable um Wert aus EEPROM zu schreiben

.equ        verz=65000                  ; Wert von dem die grosse Verzögerungsschleife herunter und wieder hochgezählt wird

            ldi     kurzverz,200
            ldi     alarmstatus,0       ; alarmstatus=0  KEIN Alarm

;-----------------------------------------------------------------------
mainloop:   wdr
;-----------------------------------------------------------------------
;Ausgänge/Eingänge
            ldi     temp,(1<<PB0|1<<PB1|1<<PB2|1<<PB3)  ; PB0 bis PB3 auf 1 ( Ausgänge)
            out     DDRB,temp

            ldi     temp,(0<<PC0 |0<<PC1 |0<<PC2); PC0 bis PC2 auf 0 (Eingänge)
            out     DDRC,temp

            ldi     temp,0
            out     DDRD,temp           ; PD2-7 = Eingänge
            sbi     PORTD,PD3           ; pullup aktivieren für PORTB3  (Alarmschleife)
            sbi     PORTD,PD4           ; pullup aktivieren für PORTB4 (Taster)
;-----------------------------------------------------------------------
;Timer/Counter und AD-Wandler
; timer wird in Unterprogramm timerstart  gestartet!!
            ldi     temp,(1<<COM20|1<<WGM21)    ;   PortB: Toggle OC2 = bei compare Match ,CTC (clear timer compare) Mode
            out     TCCR2,temp          ;       timer2 wird in Unterprogramm timerstart  gestartet!!

            ldi     temp,170
            out     OCR2,temp           ;       OutputCompareRegister2
                                        ;(bei diesem Wert wird der Timer zurückgesetzt und bestimmt die Tonhöhe)

            ldi     temp,(1<<ADLAR)|(0<<MUX1)|(0<<MUX0) ;   ADC Input:  Resultat "linksbündig" (ADC LeftAdjustResult) d.h. 8bit!
 und  PortC0
```

```
            out     ADMUX,temp          ;       in ADC Multiplexer Selection Register schreiben

;-------------------------------------------------------------------------
;Initialisierung

            rcall   leseEE              ;       Aus EEPROM Anzahl Alarmaktivierungen auslesen
            cpi     dataread,$FF        ;       wenn 255 dann wurde der Chip neu programmiert!
            brne    rese                ;       falls nicht 255  weiter zu Tasterabfrage rese
            ldi     dataread,1
            ldi     datawrite,1         ;       datawrite mit 1 füllen
            rcall   schreibeEE          ;       und in EEPROM schreiben

;Taster gedrückt? Dann Alarmzähler zurücksetzen auf 1
rese:
            SBIC    PIND,PD4            ;       PortB 4 abfragen, nächsten Befehl überspringen, wenn 1 (Taster gedrückt)
            rjmp    main                ;       Springe zu Start,weil Taster beim Einschalten NICHT gedrückt

            ldi     datawrite,1         ;       Taster gedrückt also schreibe 1 in Schreibregister
            rcall   schreibeEE          ;       Schreibregister in EEPROM
            sbi     PORTB,PB0           ;       LED-Rot EIN      Alle LED kurz einschalten als Signal
            sbi     PORTB,PB1           ;       LED-Gelb EIN    dass Speicher gelöscht
            sbi     PORTB,PB2           ;       LED-Grün EIN
            rcall pause
            cbi     PORTB,PB0           ;       LED-Rot AUS
            cbi     PORTB,PB2           ;       LED-Grün AUS
            cbi     PORTB,PB1           ;       LED-Gelb AUS

;-------------------------------------------------------------------------
            cbi     PORTB,PB0           ;       LED-Rot AUS
            cbi     PORTB,PB1           ;       LED-Gelb AUS
main:
            sbi     PORTB,PB2           ;       LED-Grün EIN
            ldi     alarmstatus,1 ;             alarmstatus auf 1
hold:
            rcall   taster              ;       Warte bis Taster gedrückt wird (=alarmstatus auf 0)
            cpi     alarmstatus,0       ;       ist alarmstatus 0?
            brne    hold                ;       nein, gehe zu hold
;-------------------------------------------------------------------------

            cbi     PORTB,PB2           ;       LED-Grün AUS:    Alarmanlage wird aktiviert
```

```
            sbi     PORTB,PB1           ;       LED-Gelb EIN
            rcall   pause
            rcall   pause
            cbi     PORTB,PB1           ;       LED-Gelb AUS

            rcall   leseEE              ;       Aus EEPROM Anzahl Alarmaktivierungen auslesen (Adresse 0x00)
                                        ;       diese werden in das Register dataread kopiert
;---------------------------------------------------------------------------
            ldi kurzverz,100            ;       kurzverz bestimmt die Länge der Verzögerung im Unterprogramm "pause"


blink:                                  ;       Anzahl Alarmaktivierungen durch Blinken anzeigen
            dec     dataread            ;       dataread=dataread-1
            cpi     dataread,0          ;       ist dataread=0 ??
            breq    start               ;       wenn ja - springe zu start

                                        ;Anzahl Blinks = Anzahl Alarmaktivierungen
            cbi     PORTB,PB0           ;       LED EIN
            rcall   pause               ;       Blinkverzögerung
            sbi     PORTB,PB0           ;       LED AUS
            rcall   pause               ;       Blinkverzögerung
            rjmp    blink               ;       nochmals von vorn
;---------------------------------------------------------------------------
start:                                  ;       ALARMANLAGE AKTIV
            clr     temp                ;       Variable Temp löschen
            out     TCCR2,temp          ;       Timer stoppen >> kein Ton
            cbi     PORTB,PB0           ;       LED-rot AUS

                                        ;       Status PIND in Variable temp kopieren
            sbic    PIND,PIND3          ;       ist PIN3 in PORTD low(Alarmschleife zw. Masse und Pin3 NICHT unterbrochen??)
            rjmp    alarm               ;       wenn unterbrochen (bit set) dann springe zu Alarm

            rcall   leseADC2            ;       Lichtwert von LDR via ADC an PORTB,2 einlesen (und in Register licht speichern)
            ldi     temp,100
            cp      licht,temp          ;       ist "licht"-Wert kleiner 100?
            brlo    start               ;       wenn ja dann springe zurück zu start:  (kein Licht auf dem LDR)
;---------------------------------------------------------------------------
alarm:
            rcall   leseEE              ;       lese EEPROM Adresse 0 in dataread ein
            inc     dataread            ;       Alarmzähler erhöhen
```

```
            mov     datawrite,dataread  ;       in Datenausgangsregister schreiben
            rcall   schreibeEE          ;       und in EEPROM Adresse 0x0 hineinschreiben

alarmrepeat:
            ldi     alarmstatus,1       ;       Alarm ausgelöst
            rcall   leseADC             ;       Potentiometereinstellung für Sirenengeschwindigkeit einlesen
            rcall   timerstart          ;       Timer starten, Ton ein
            cbi     PORTB,PB0           ;       LED-Rot AUS
            rcall   pause               ;       Blinkverzögerung und gleichzeitig Tonhöhe verändern
            rcall   leseADC

            sbi     PORTB,PB0           ;       LED-Rot EIN

            rcall   pause
            rcall   leseADC

            cpi     alarmstatus,1       ;       Wenn Alarm
            breq    alarmrepeat         ;       springe zu Alarm wiederholung
            rjmp    start               ;       sonst zu start
;-----------------------------------------------------------------------------

;TIMER STARTEN
timerstart:
            ldi     temp,(1<<CS21|1<<COM20|1<<WGM21 )   ;timer start, prescaler 8
            out     TCCR2,temp          ;       TimerCntrCntRlRegister 0 B
            ret
;-----------------------------------------------------------------------------
; Verzögerung und Sirenenschleife
pause:
            ldi     ZH,HIGH(verz)       ;       lade doppelregister Z mit Zahl "verz"
            ldi     ZL,LOW(verz)
again1:                                 ;       Ab hier Doppelvariable herunterzählen und
                                        ;       High Byte in OCR2 kopieren  (ZH wird immer kleiner, der Ton damit immer höher)
            out     OCR2,ZH             ;       OutputCompareRegister 2 ( bei diesem Wert wird der Timer zurückgesetzt)
            rcall   taster              ;       Ist der Taster gedrückt? (Alarm reset)
            rcall   kurzepause
            rcall   taster

            sbiw    ZL,32               ;       doppelreg. Z um 32 erniedrigen (low byte)
            cpi     ZH,70               ;       ist ZH unter 70?
```

```
            brlo    next                ;       weiter wenn Doppelreg.HighByte <70(damit Ton nicht zu hoch wird: OCR2 nicht <70)
            rjmp    again1              ;       ZH noch > 70 also nochmal
next:

again2:                                 ;       Ab hier Doppelregister Z wieder erhöhen bis zur Konstante "verz"
            rcall   taster
            rcall   kurzepause
            rcall   taster              ;       Wieder High Byte in OCR2 kopieren

                                        ;      damit wird der Ton wieder tiefer, weil der Timer später zurücksetzt
            out     OCR2,ZH             ;       OutputCompareRegister 2 ( bei diesem Wert wird der Timer zurückgesetzt)

            adiw    ZL,32               ;       doppelreg. Z um 32 erhöhen (low byte)
            cpi     ZH,HIGH(verz)       ;       vergleiche hochgezählten Wert
                                        ;       ( HighByte des Doppelregisters) mit der oberen Verzögerungsgrenze
            brne    again2
            ret
;-----------------------------------------------------------------------------------------
kurzepause:
            push    temp
            mov     temp,kurzverz       ;       kleine Verzögerungsschleife Dauer gem. Register kurzverz (eingelesen von Poti vi
a AD-Wandler)
repeat:
            dec     temp

            brne    repeat
            pop     temp
            ret
;-----------------------------------------------------------------------------------------
;Abfrage ob Taster gedrückt (PIN auf LOW = gedrückt
taster:
            SBIS    PIND,PD4            ;       PortB 4 abfragen, nächsten Befehl überspringen, wenn 1 (Taster NICHT gedrückt)
            ldi     alarmstatus,0       ;       Setze Alarmstatus auf 0
            ret
;-----------------------------------------------------------------------------------------
; Potentiometer zur Einstellung der Sirenengeschwindigkeit einlesen
leseADC:
            ldi     temp,(1<<REFS1|1<<REFS0|1<<ADLAR)|(0<<MUX0)|(0<<MUX1);REFerenceSelectionBits0 + 1: int.Ref. ,
                                        ;Input:Resultat "linksbündig" (ADcLeftAdjustResult) d.h. 8bit! u. PortC0 als Input auswä
hlen
```

```
            out     ADMUX,temp          ; in ADC Multiplexer Selection Register schreiben

            ldi     temp,(1<<ADEN)|(1<<ADSC)|(1<<ADPS2)|(1<<ADPS0)  ;
                                        ;ADENable, ADStartConversion, ADPrescalerSelect 2 und 0: Teiler=32
            out     ADCSRA,temp         ; Abfragetakt (muss 50-200kHz sein) = CPU Frequenz dividiert durch 32
warteADC:                               ;   3.6MHz/32=112kHz
            sbic    ADCSRA,ADSC         ; Warte bis ADC  fertig (überspringe nächsten Befehl wenn Bit ADSC wieder 0)
            rjmp    warteADC
            in      kurzverz,ADCH       ; Lese oberes 8-Bit des AD-Wandlers und schreiben Resultat in "Geschwindigkeitsvariable"
            cpi     kurzverz,0          ; wenn kurzverz 0
            brne    label
            ldi     kurzverz,1          ; dann kurzverz =1
label:
            ret
;-----------------------------------------------------------------------------------------------
; LDR zur Lichtmessung einlesen
leseADC2:
            ldi     temp,(1<<REFS1|1<<REFS0|1<<ADLAR)|(1<<MUX0)|(0<<MUX1);REFerenceSelectionBits0 + 1: int.Ref. ,
                                        ;Input:Resultat "linksbündig" (ADC LeftAdjustResult) d.h.8bit! u.PortC1 als Input wählen
            out     ADMUX,temp          ;in ADC Multiplexer Selection Register schreiben

            ldi     temp,(1<<ADEN)|(1<<ADSC)|(1<<ADPS2)|(0<<ADPS1)|(1<<ADPS0)

; ADENable, ADStartConversion, ADPrescalerSelect 2 und 0: Teiler=32
            out     ADCSRA,temp         ; Abfragetakt (muss 50-200kHz sein) = CPU Frequenz dividiert durch 32
warteADC2:                              ;   3.6MHz/32=112kHz
            sbic    ADCSRA,ADSC         ; Warte bis ADC  fertig (überspringe nächsten Befehl wenn Bit ADSC wieder 0)
            rjmp    warteADC2
            in      licht,ADCH          ; Lese oberes 8-Bit des AD-Wandlers Kopiere Resultat in "Lichtvariable"
            cpi     licht,0             ; wenn licht 0
            brne    label2
            ldi     licht,1             ; dann licht =1
label2:
            ret
;-----------------------------------------------------------------------------------------------
;Schreibe Inhalt des Registers "datenein" in EEPROM (Adresse 0x00)
schreibeEE:

warteEE:
            sbic    EECR,EEWE           ; EEpromControlRegister EEWriteEnable
```

```
            rjmp    warteEE             ; warte bis EEPROM bereit
            ldi     XL,0x00             ;Adresse EEPROM Zelle hex 0
            ldi     XH, 0x00

            out     EEARH,XH            ;Adresse in EEPR.-AdressRegisterHigh schreiben
            out     EEARL,XL            ;Adresse in EEPR.-AdressRegisterLow schreiben

            out     EEDR,datawrite      ;zahl (in register daten) in Datenregister schreiben
            sbi     EECR,EEMWE          ;Enable bit setzen (EE MasterWriteEnable)
            sbi     EECR,EEWE           ;setze Write Enable
            ret
;-----------------------------------------------------------------------------------------------
;Lese Inhalt des EEPROM (Adresse 0x00) und kopiere in Register "datenaus"
leseEE:

            ldi     XL,0x00             ;Adresse EEPROM Zelle hex 0
            ldi     XH, 0x00
            out     EEARL,XH                ;Adresse in EEPR.-AdressRegisterHi schreiben
            out     EEARL,XL                ;Adresse in EEPR.-AdressRegisterLow schreiben
            sbi     EECR,EERE           ;ReadEnableBit in EE-ControlRegister
            in      dataread,EEDR       ;Inhalt EEDataRegister in Register datenaus schreiben
            ret
;-----------------------------------------------
            rjmp    mainloop
;-------------------------------------------------------------------
```