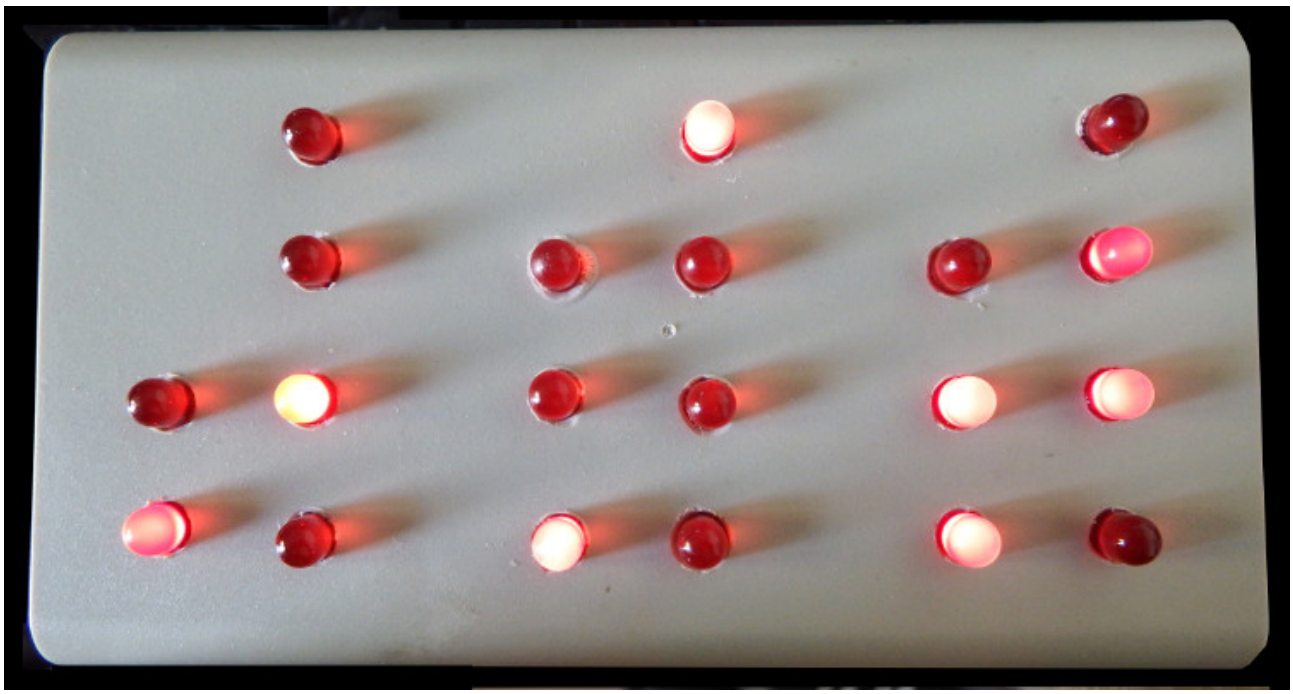
Benutzanleitung fur

Duenne-Berg-Uhr



von

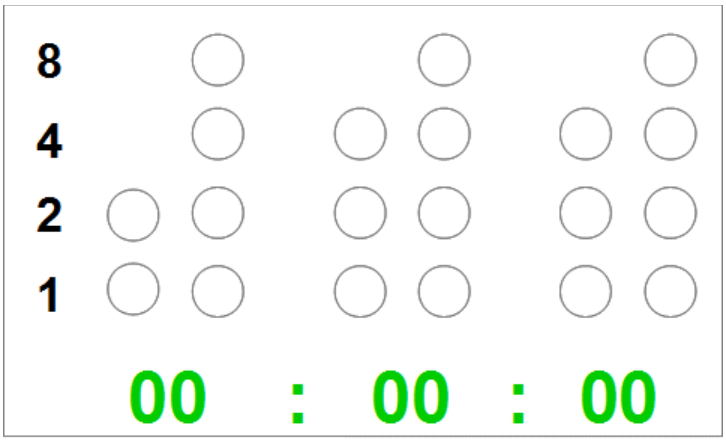
Schmidt Microsystems
Darmstadt/Germany
http://www.avr-asm-tutorial.net/avr_de/duenne-berg-uhr/

Mai 2017

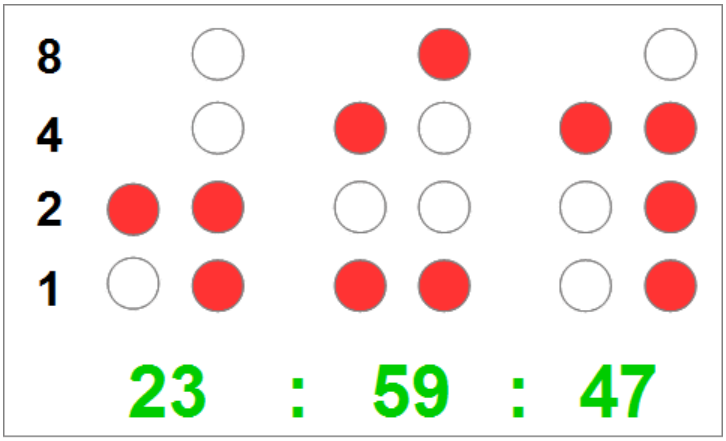
Herzliche Gluckwunsch
fur Erwerb von Duenne-Berg-Uhr komplett!

Sie haben damit hochmoderne Mikroprofessor-Uhr mit ganz genaue Gang, weil hat Schwingerquarz mit genau 2.465.700 Hertz und macht genaues Teilen von das.

Das ist keine Dicke-Tal-Uhr, aber Duenne-Berg-Uhr, weil hat Berg aus Leuchtdioden rot von 20 Stuck in duennes Gehause. Dicke-Tal-Uhren zeigen Zeit auf Sieben-Leuchten an, wo sehen kann Zahlen. Duenne-Berg-Uhr alles anders: wenn nachts 24 Uhr, alle Leuchten aus:



Duenne-Berg-Uhr zeigt Zeit mit Kopfrechnen an: jede Leuchte zwei mal mehr zaehlt als niedrige wie bei Computer. Kannst sehen in Bild:



Musst zaehlen Zahlen zusammen in Kopf gleiche Spalte fur Uhrzeitlesen.

Uhr hat nix fur Stellen Zeit. Fur Stellen Uhr schalten Uhr eine Minute vor zwolf aus und genau zwolf wieder an. Dann Uhr geht ganz genau bis Stromabstellung.

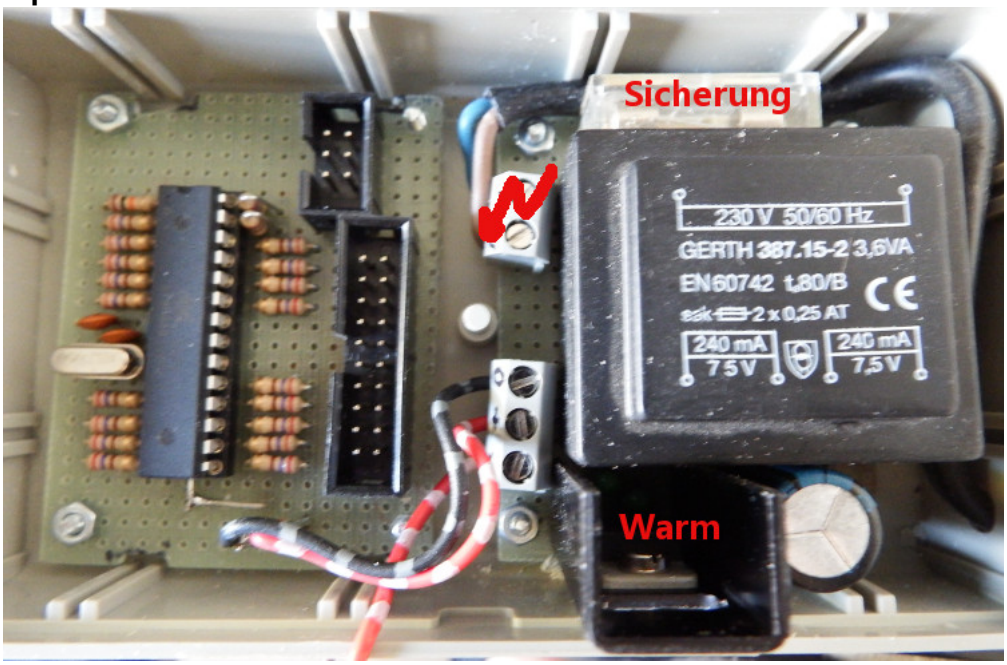
Uhr hat nix Wecker, so kannst schlafen bis Puppen.

Stromversorgung:

Strom kommt aus Steckdose 230 Volte. Verbrauch ist maximal 10 Kilowattstunde wenn ganzes Jahr an, macht Geld drei Euro.

Repariere:

Kannst offnen Schachtel fur Reparieren. Musst aber ziehen Stecker von Strom vorher. Gerat hat Sicherung vor zu viel Strom, kannst rausholen und anderes reinmachen (musst kaufen Sicherung 100 mA langsam). Sonst nix fur Reparieren in Schachtel.



Kwallidaed:

Hat getestet Dauerstress mit Stromverbrauch (300 mA ueber sechs Stunden heiss bei normal 150 mA): nix Ausfall.
Hat getestet Software mit Schnellmethode (1 Minute in 1 Sekunde, 1 Stunde in 1 Sekunde): alles richtig anzeigen.
Nix ISO9000, nix Euro-Prufsiegel, bix TUV, weil sowieso nur Schmuh fur dumm Leut, aber Gerat solide gebaut und beachten Sicherheit.

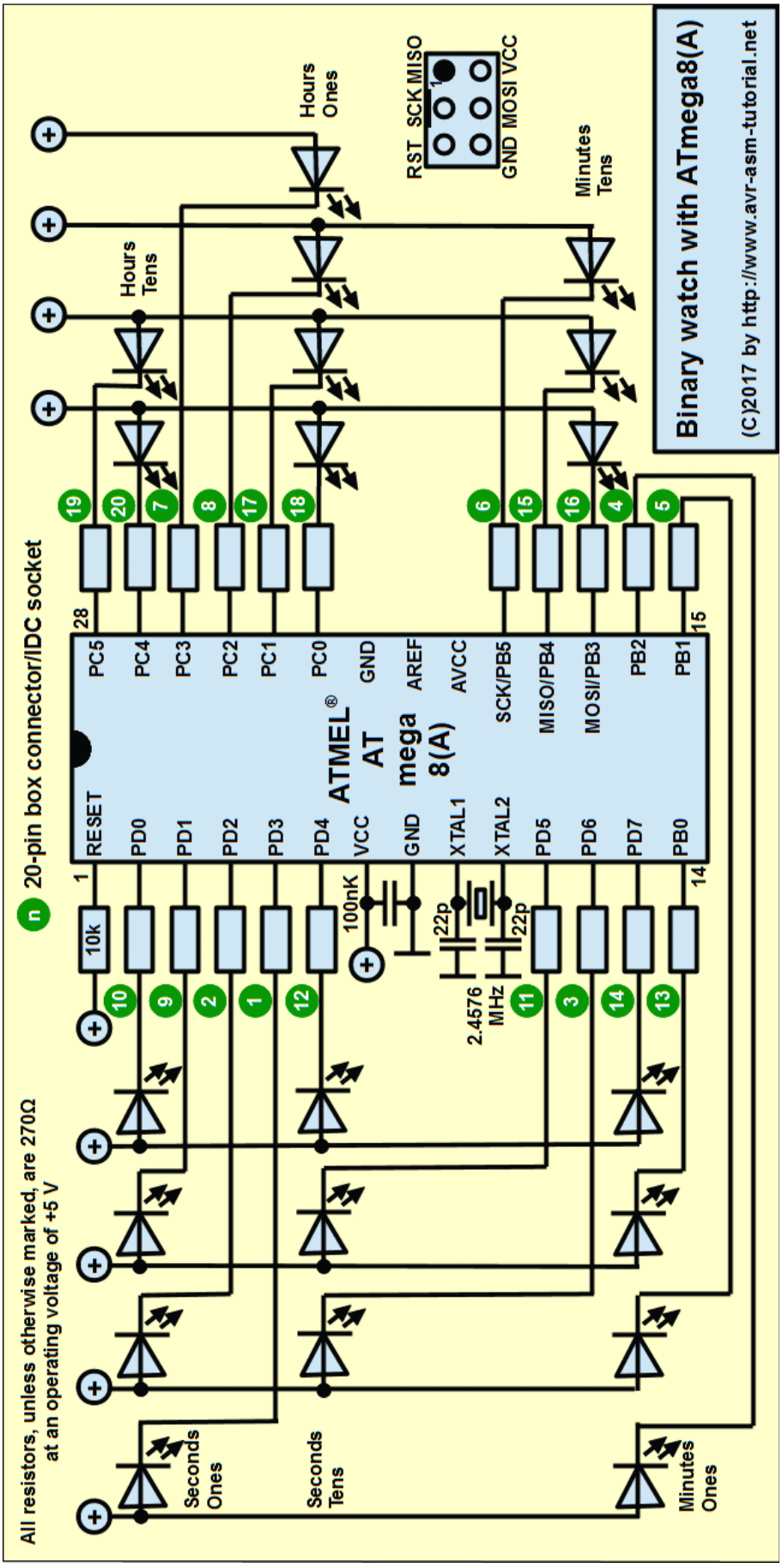
Garantie:

Hast zwei Jahre Garantie auf richtig funktionieren. Wenn kaputt musst anrufen Hersteller auf kostenpflichtig Servicenummer (0900-1234567, kost 100 Euro pro Minut). Reparieren kostenlos, aber nur wenn nicht geschmissen vorher in Badewanne oder aus Flugzeug.

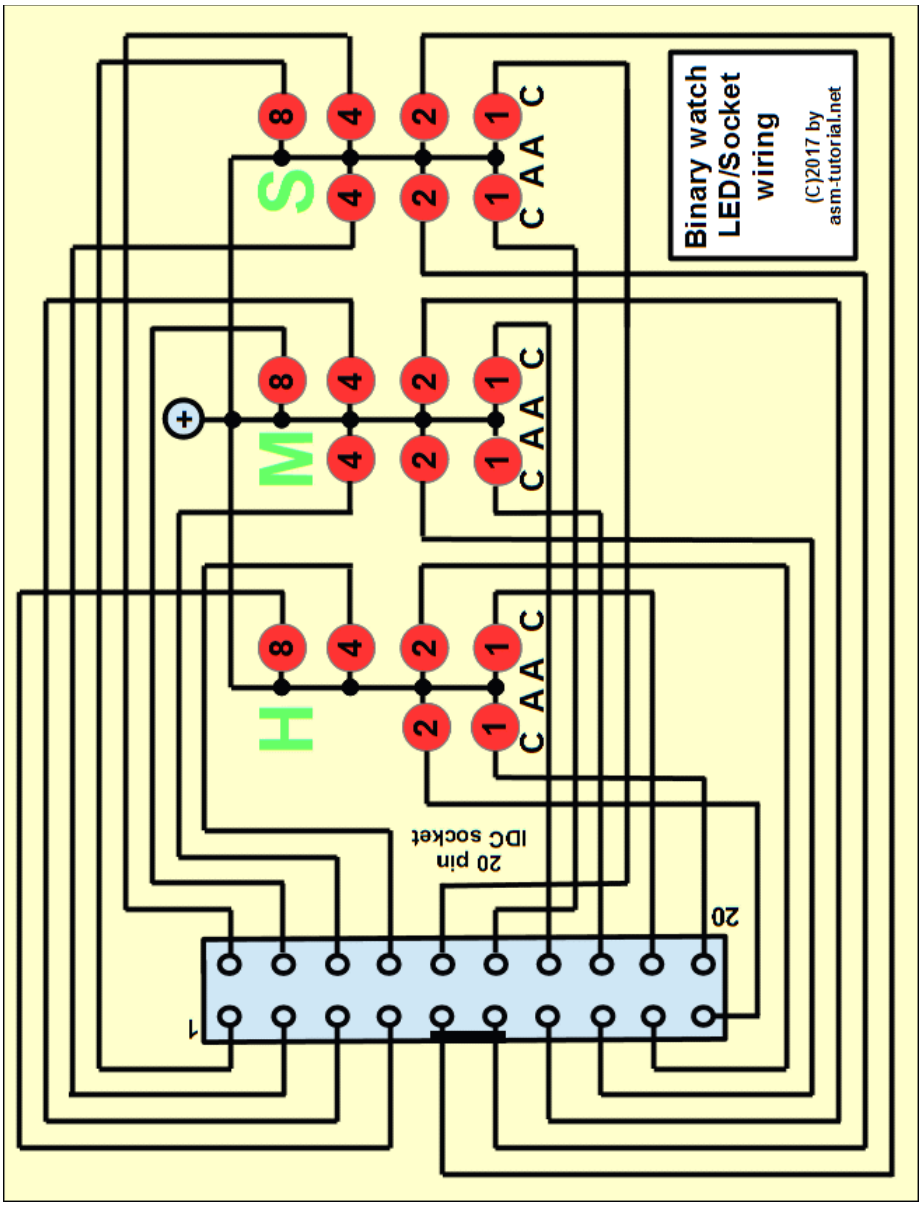
Anhang:

1. Schaltbild von Geraet Professor
2. Schaltbild von Lampen und Stecker
3. Schaltbild von Stromteil
4. Layout von Geraet
5. Programm von Mikroprofessor (Assembler)

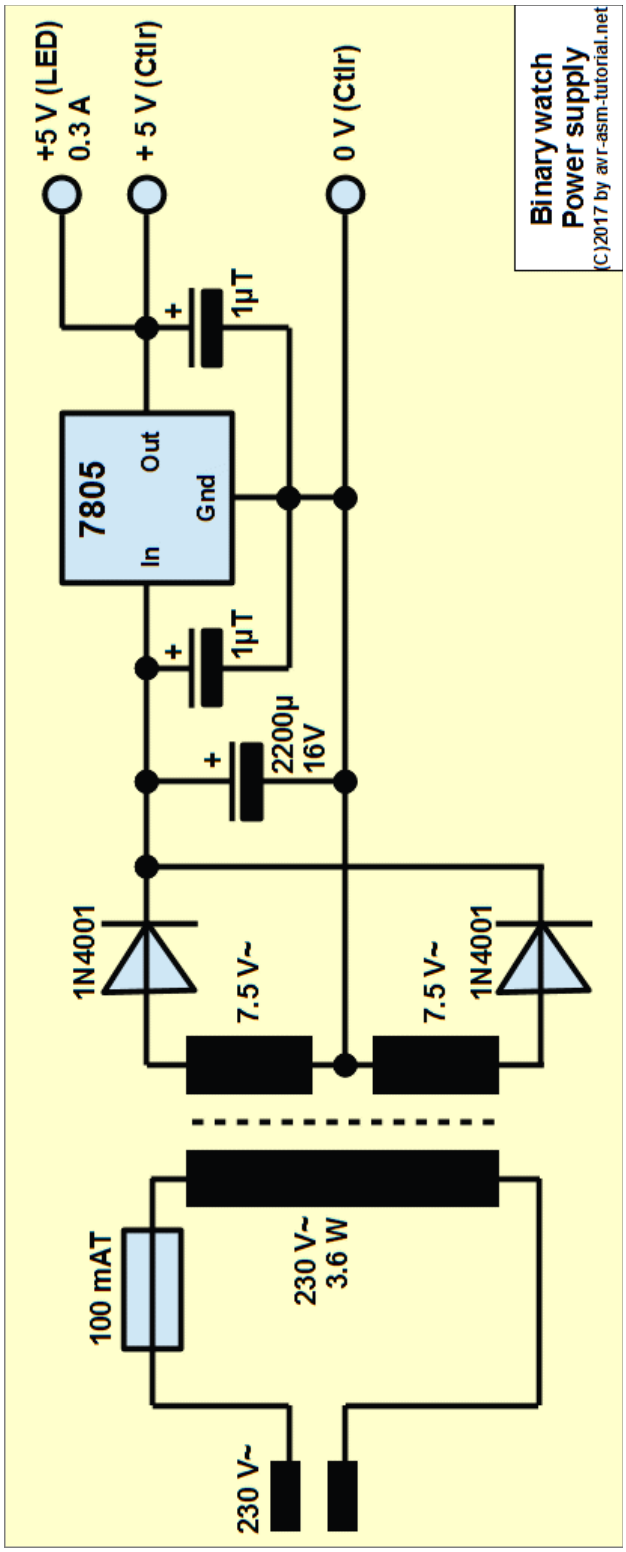
Anhang 1: Schaltbild von Geraet Professor



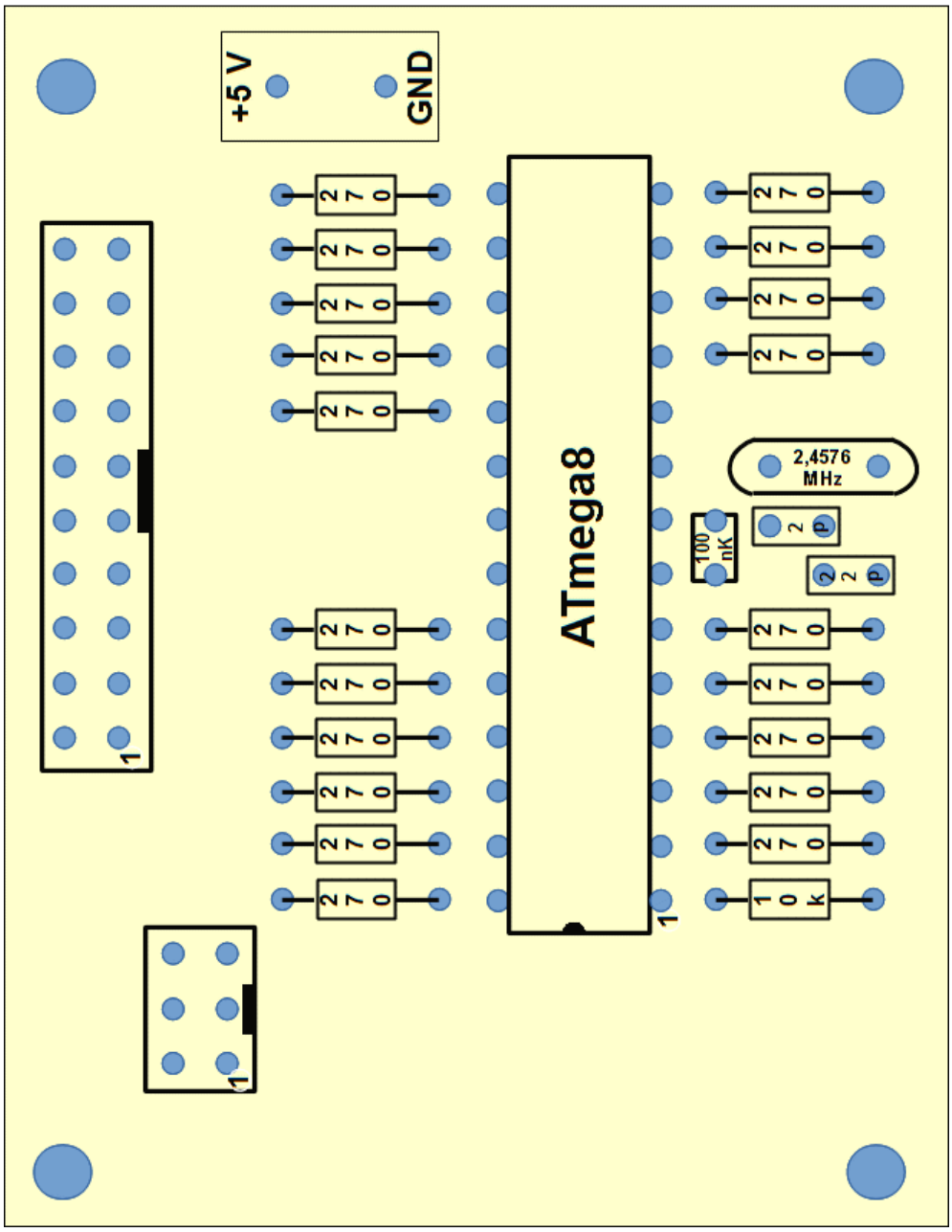
Anhang 2: Schaltbild von Lampen und Stecker



Anhang 3: Schaltbild von Stromteil



Anhang 4: Layout von Geraet



Anhang 5: Programm von Mikroprofessor (Assembler)

```
;
; ******************************************
; * Binaere Uhr mit 20 LEDs und ATmega8A   *
; * Version 1, April 2017                  *
; * (C)2017 by http://www.avr-asm-tutorial.net *
; ******************************************
;
.NOLIST
.INCLUDE "m8Adef.inc" ; Headerdatei fuer ATMEGA8A
.LIST
;
; ====================================
;  A S S E M B L E R - S C H A L T E R
; ====================================
;
; Diese Assemblerschalter fuer normalen Ablauf = 0
.equ speed = 0 ; 0 = normal, 1 = Minuten testen, 2 = Stunden testen
.equ fast = 0 ; Schneller Ablauf fuer Debug (Prescaler=1)
;
; Dieser Schalter ist Geschmackssache, pbcd=1 ist schneller
.equ pbcd = 1 ; 1 = Uhrzeit in Packed BCD, 0 = Binaer
;
; ==========================================
;   H A R D W A R E I N F O R M A T I O N E N
; ==========================================
;
; ATmega8A
;                  _________________
;  ISP6+   1 /                 |28
;  RESET o--|reset          PC5|--o StdZ2
;          2|                  |27
;  SekE1 o--|PD0            PC4|--o StdZ1
;          3|                  |26
;  SekE2 o--|PD1            PC3|--o StdE8
;          4|                  |25
;  SekE4 o--|PD2            PC2|--o StdE4
;          5|                  |24
;  SekE8 o--|PD3            PC1|--o StdE2
;          6|                  |23
;  SekZ1 o--|PD4            PC0|--o StdE1
;          7|                  |22
;   +5 V o--|vcc            gnd|--o NC
;          8|                  |21
;    0 V o--|gnd           AREF|--o NC
;          9|                  |20
;  Quarz o--|XTAL1         AVCC|--o NC
; 2,4576  10|                  |19
;   MHz  o--|XTAL2      PB5/SCK|--o MinZ4+ISP6-SCK
;         11|                  |18
;  SekZ2 o--|PD5      PB4/MISO|--o MinZ2+ISP6-MISO
;         12|                  |17
;  SekZ4 o--|PD6      PB3/MOSI|--o MinZ1+ISP6-MOSI
;         13|                  |16
;  MinE1 o--|PD7           PB2|--o MinE8
;         14|                  |15
;  MinE2 o--|PB0           PB1|--o MinE4
;           |_________________|
;
;
```

```
; ================================================
; K O N S T A N T E N   Z U M  E I N S T E L L E N
; ================================================
;
.equ clock = 2457600 ; Taktfrequenz, externer Quarz
;
; ========================================================
;  F E S T E + A B G E L E I T E T E  K O N S T A N T E N
; ========================================================
;
.if speed == 0 ; normale Geschwindigkeit
  .equ cPresc = 1024 ; Teiler fuer Timer 1
  .equ cCtc = clock/cPresc - 1 ; Teiler Timer 1
  .elif speed == 1 ; Minutentest
  .equ cPresc = 64
  .equ cCtc = clock/cPresc/60 - 1
  .else ; Stundentest
  .equ cPresc = 1
  .equ cCtc = (clock/cPresc+1800)/3600 - 1
  .endif
;
; ============================================
;   R E G I S T E R D E F I N I T I O N E N
; ============================================
;
; frei: R0 bis R15
.def rmp1 = R16 ; Vielzweckregister 1
.def rmp2 = R17 ; Vielzweckregister 2
.def rSek = R18 ; Sekunden
.def rMin = R19 ; Minuten
.def rStd = R20 ; Stunden
.def rPD = R21  ; Portausgabe Register Port D
.def rPB = R22  ; dto., Port B
.def rPC = R23  ; dto., Port C
; frei: R24 bis R31
;
; ============================================
;       S R A M   D E F I N I T I O N E N
; ============================================
;
.DSEG
.ORG  0X0060
; (SRAM nur fuer Interrupt und Call-Stack verwendet)
;
; ============================================
;   R E S E T   U N D   I N T   V E K T O R E N
; ============================================
;
.CSEG
.ORG $0000
      rjmp Main ; Reset-Vektor
      reti ; INT0
      reti ; INT1
      reti ; TC2COMP
      reti ; TC2OVF
      reti ; TC1CAPT
      rjmp SekInt ; TC1COMPA, Sekundeninterrupt
      reti ; TC1COMPB
      reti ; TC1OVF
      reti ; TC0OVF
      reti ; SPI_STC
      reti ; UART-RX
      reti ; UART-UDRE
      reti ; UART-TXC
```

```
      reti ; ADC
      reti ; EERDY
      reti ; ANACOMP
      reti ; TWI
      reti ; SPMRDY
;
; =========================================
;    I N T E R R U P T   S E R V I C E
; =========================================
;
; Sekunden-Interrupt
; Da keine anderen Interrupts und keine
;   weiteren Ablaeufe zu bearbeiten sind,
;   erfolgt der gesamte Ablauf innerhalb
;   der Interrupt-Service-Routine
SekInt: ; Sekunden-Interrupt
      inc rSek ; naechste Sekunde
.if pbcd == 1
; Packed-BCD
      ldi rmp1,0x06 ; fuer Packed-BCD-Halbuebertrag
      add rSek,rmp1 ; Halbuebertrag?
      brhs SekInt1 ; Ja, teste Sekunden=60
      sub rSek,rmp1 ; kein Halbuebertrag, ziehe wieder ab
      rjmp SekInt4 ; fertig zur Ausgabe
SekInt1:
      cpi rSek,0x60 ; 60 Sekunden?
      brcs SekInt4 ; Nein, zur Ausgabe
      clr rSek ; Sekunden = 0
      inc rMin ; naechste Minute
      add rMin,rmp1 ; Halbuebertrag?
      brhs SekInt2 ; ja, teste 60 Minuten
      sub rMin,rmp1 ; Nein, wieder abziehen
      rjmp SekInt4 ; Zur Ausgabe
SekInt2:
      cpi rMin,0x60 ; 60 Minuten?
      brcs SekInt4 ; Nein, zur Ausgabe
      clr rMin ; Minuten = 0
      inc rStd ; naechste Stunde
      add rStd,rmp1 ; Halbuebertrag?
      brhs SekInt3 ; Ja, teste Stunden=24
      sub rStd,rmp1 ; nein, ziehe wieder ab
      rjmp SekInt4 ; zur Ausgabe
SekInt3:
      cpi rStd,0x24 ; 24 Stunden?
      brcs SekInt4 ; nein, zur Ausgabe
      clr rStd ; Stunden auf Null
SekInt4: ; Ausgabe vorbereiten
      mov rPC,rStd ; Port C sind die Stunden
      mov rPD,rSek ; Port D sind die Sekunden
      lsl rPD ; oberstes Bit herausrollen
      mov rPB,rMin ; Port B sind die Minuten
      lsr rPB ; unterstes Bit herausrollen
      ror rPD ; und in oberstes Bit Port D hinein
      com rPD ; Portbits umkehren wegen LED-Polaritaet
      com rPB
      andi rPB,0x3F ; oberste Bits maskieren
      com rPC
      andi rPB,0x3F ; oberste Bits maskieren
      ; Ausgabe an Ports
      out PORTD,rPD
      out PORTB,rPB
      out PORTC,rPC
      reti ; fertig
```

```
      .else
      ; Binaere Uhrzeit in drei Registern
      cpi rSek,60 ; 60 Sekunden?
      brcs SekInt1 ; nein, Ausgabe
      clr rSek ; ja, Sekunden auf Null
      inc rMin ; naechste Minute
      cpi rMin,60 ; 60 Minuten?
      brcs SekInt1 ; nein, Ausgabe
      clr rMin ; ja, Minuten auf Null
      inc rStd ; naechste Stunde
      cpi rStd,24 ; 24 Stunden?
      brcs SekInt1 ; nein, Ausgabe
      clr rStd ; ja, Stunden auf Null
SekInt1: ; Ausgabe vorbereiten
      mov rmp1,rSek ; Sekunden in Register
      rcall Div10 ; durch 10 dividieren
      mov rPD,rmp1 ; Zehnerrest in Portregister D
      swap rmp2 ; Zehner in oberes Nibble
      or rPD,rmp2 ; Oberes und unteres Nibble kombinieren
      ; Minuten
      mov rmp1,rMin ; Minuten in Register
      rcall Div10 ; Division durch 10
      lsr rmp1 ; Einer rechts schieben
      brcc SekInt2 ; Kein Carry, Sekundenregister ok
      sbr rPD,0x80 ; setze oberstes Bit Sekundenregister
SekInt2:
      lsl rmp2 ; Schiebe Zehner drei Mal links
      lsl rmp2
      lsl rmp2
      or rmp1,rmp2 ; Kombiniere mit unteren Bits
      mov rPB,rmp1 ; und in Portregister B
      ; Stunden
      mov rmp1,rStd ; Stunden in Register
      rcall Div10 ; Division durch 10
      swap rmp2 ; Zehner in oberes Nibble
      or rmp1,rmp2 ; oberes und unteres Nibble kombinieren
      mov rPC,rmp1 ; und in Portregister C
      ; Invertieren vor der Ausgabe wegen LED-Polaritaet
      com rPD
      com rPB
      andi rPB,0x3F ; obere Bits ausblenden
      com rPC
      andi rPC,0x3F ; obere Bits ausblenden
      ; Portregister auf Ports ausgeben
      out PORTD,rPD
      out PORTB,rPB
      out PORTC,rPC
      reti
;
; Dividiere rmp1 durch 10
; Ergebnis in rmp2, Rest in rmp1
Div10:
      clr rmp2 ; Divisionsergebnis leeren
Div10a:
      subi rmp1,10 ; Zehn abziehen
      brcs Div10b ; Carry, fertig
      inc rmp2 ; Ergebnis erhoehen
      rjmp Div10a ; und weiter dividieren
Div10b:
      subi rmp1,-10 ; 10 zum Rest wieder addieren
      ret ; fertig
      .endif
```

```
;
; =========================================
;    H A U P T P R O G R A M M    I N I T
; =========================================
;
Main:
; Initiiere Stapel
      ldi rmp1, HIGH(RAMEND) ; Initiiere MSB Stapel
      out SPH,rmp1
      ldi rmp1, LOW(RAMEND) ; Initiiere LSB Stapel
      out SPL,rmp1
; Init PORT D
      ldi rmp1,0xFF ; alle als Ausgang
      out DDRD,rmp1
      out PORTD,rmp1
; Init Port B
      ldi rmp1,0x3F ; alle als Ausgang
      out DDRB,rmp1
      out PORTB,rmp1
; Init Port C
      out DDRC,rmp1 ; alle als Ausgang
      out PORTC,rmp1
; Init Uhrzeit, Stellen der Uhr auf 12:00:00
.if pbcd == 1
      ldi rStd,0x12 ; Packed-BCD-Uhrzeit
      ldi rMin,0x00
      ldi rSek,0x00
      .else
      ldi rStd,12 ; Binaer-Uhrzeit
      ldi rMin,0
      ldi rSek,0
      .endif
; Init Timer 1
      ldi rmp1,High(cCtc) ; CTC-A-Wert setzen
      out OCR1AH,rmp1
      ldi rmp1,Low(cCtc)
      out OCR1AL,rmp1
      ldi rmp1,0 ; Timer-Mode Normal/CTC-A
      out TCCR1A,rmp1
.if (fast == 1) | (speed == 2) ; bei Schnell und bei Stundentest
      ldi rmp1,(1<<WGM12)|(1<<CS10) ; Prescaler = 1
      .elif speed == 0 ; Normalbetrieb
      ldi rmp1,(1<<WGM12)|(1<<CS12)|(1<<CS10) ; normal, Prescaler=1024
      .elif speed == 1
      ldi rmp1,(1<<WGM12)|(1<<CS11)|(1<<CS10) ; Prescaler=64
      .endif
      out TCCR1B,rmp1 ; Timer Mode CTC-A, Prescaler
      ldi rmp1,1<<OCIE1A ; Compare-A-Interrupt
      out TIMSK,rmp1 ; in Timer-Interrupt Maske
;
; Sleep und Interrupts
      ldi rmp1,1<<SE ; Enable sleep mode idle
      out MCUCR,rmp1 ; in MCU-Kontrollport
      sei ; Enable Interrupts
;
; =========================================
;     P R O G R A M M - S C H L E I F E
; =========================================
;
Loop:
      sleep ; Schlafen legen
      nop ; Dummy fuer Aufwecken
      rjmp loop ; Zurueck nach Loop
;
```

```
; Ende Quellcode
; Copyright
.db "(C)2017 by Gerhard Schmidt  " ; menschenlesbar
.db "C(2)10 7ybG ehrra dcSmhdi t " ; wortgerecht
;
```