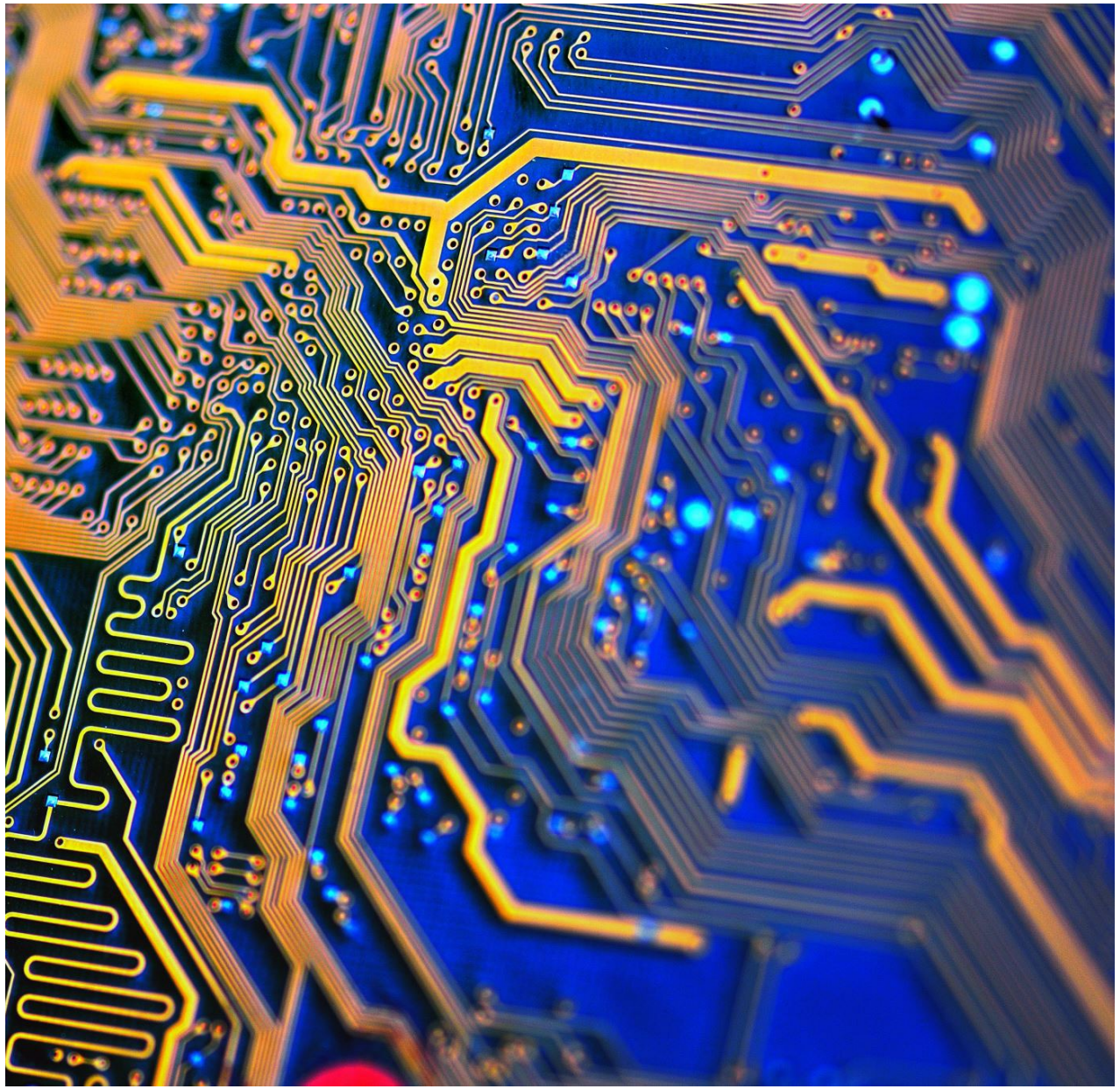
Systemnahe Programmierung Hardware

Binäruhr

HTWK Leipzig



Setup

Global variables

```
// Global variables for setting Timer Hours, Minutes and Seconds
volatile uint8_t hours = 3;
volatile uint8_t minutes = 16;
volatile uint8_t seconds = 0;
volatile uint8_t ledSeq[11] = {0};

// Global variable for debouncing button and sleep modus
volatile int bouncer = 0;
volatile int sleep = 0;
```

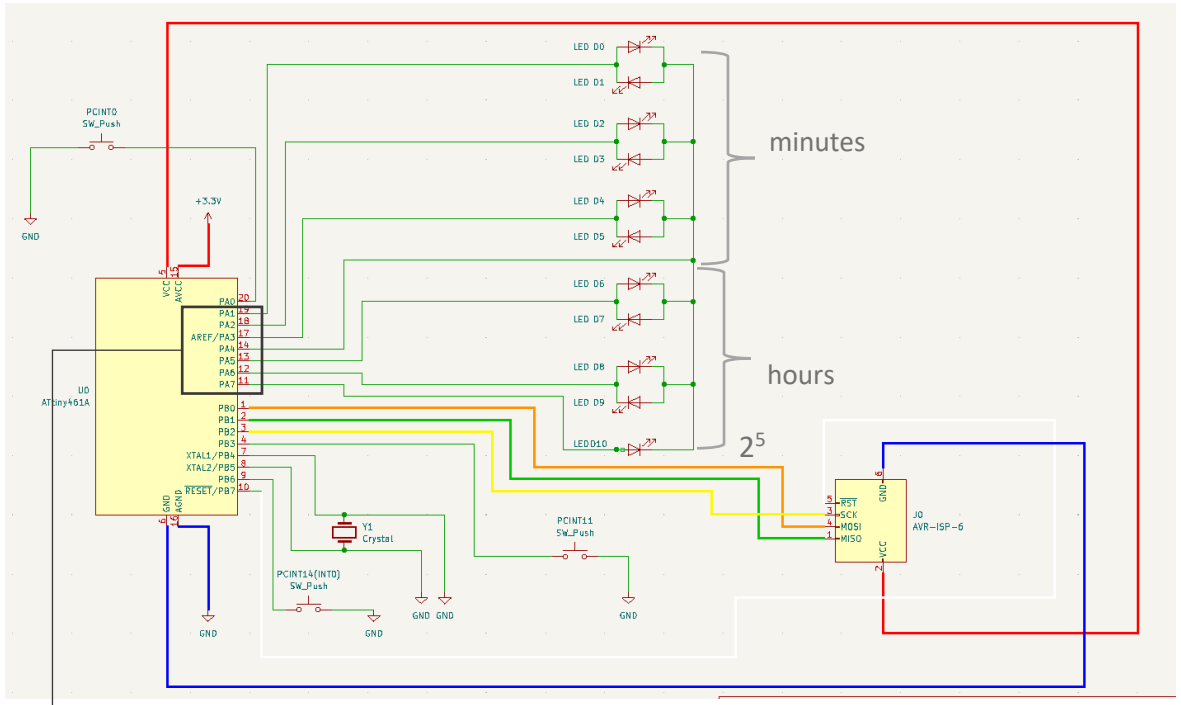
sleep++ in ISR(timer)

bouncer++ in main()

*ISR – Interrupt Service Routine

Setup

PIN Configuration



LED Pins

```
DDRA = 0b11111110; // Switch Reg A to 1-7 output and 0 to input
DDRB = 0b00000000; // Switch Reg B to INPUT (Pin PB6 & PB3 are Interrupts)
```

```
// constructing led pairs (LED to be shown, Pin-config for multiplexing)
struct pair
{
    uint16_t bin;
    uint8_t ledPins;
};

/*
   initialize pair array
const struct pair ledPairs[] =
{
        bin          ledPins
    {0b00000000001, 0b00000011},    //D0
    {0b00000000010, 0b01111101},    //D1
    {0b00000000100, 0b00000101},    //D2
    {0b00000001000, 0b01111011},    //D3
    {0b00000010000, 0b00001001},    //D4
    {0b00000100000, 0b01110111},    //D5
    {0b00001000000, 0b00100001},    //D6
    {0b00010000000, 0b01011111},    //D7
    {0b00100000000, 0b01000001},    //D8
    {0b01000000000, 0b00111111},    //D9
    {0b10000000000, 0b10000001}     //D10
};
```

PCINT0
HIGH

A 7 6 5 4 3 2 1 0

0b 100 0 000 1

A7 == HIGH
= D10 High

hours minutes

2^5 .... 2^0 2^6 ..... 2^0

0b 10000 000000

2^5 Uhr
= 16 Uhr

hours minutes

Setup

Timer

```
void setup()
{
    DDRA = 0b11111110; // Switch Reg A to 1-7 output and 0 to input
    DDRB = 0b00000000; // Switch Reg B to INPUT (Pin PB6 & PB3 are Interrupts)

    TCCR1B |= (1 << CS13); //set Timer1(Time) auf f/128 bei f= 2^15 overflow nach 1s
    TIMSK |= (1 << TOIE1); // set interrupts for Timer1

    //Pin Change Interrupts - enable PCINT0, PCINT11 und PCINT14
    //Pull up Widerstände setzen
    PORTA |= (1 << PORTA0);
    PORTB |= (1 << PORTB6) | (1 << PORTB3);

    GIMSK |= (1 << PCIE1) | (1<< PCIE0); //enable General Interrupt Mask Register for PCINT[7:0] or PCINT[15:12]
    PCMSK0 |= tastHours; //enable PCINT0 on PORT A PIN A0
    PCMSK1 |= tastMinSleep; //enable PCINT14 und PCINT11 on PORTB PIN B6

    sei(); // enable global interrupts same as SREG |= 0x80;

    /*
        remains enabled: timers, pins pullups, interrupts, Clock, OSC
        disables: CLK_cpu, CLK_flash aka no new instructions are pulled prom flash nor are they processed
    */
    set_sleep_mode(SLEEP_MODE_IDLE);

    //inits LEDs according to initiated global minutes and hours
    updateSeq();
}
```

Timer/Counter 1
Overflow enable
ISR(overflow)

TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x2F (0x4F)	PWM1X	PSR1	DTPS11	DTPS10	CS13	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

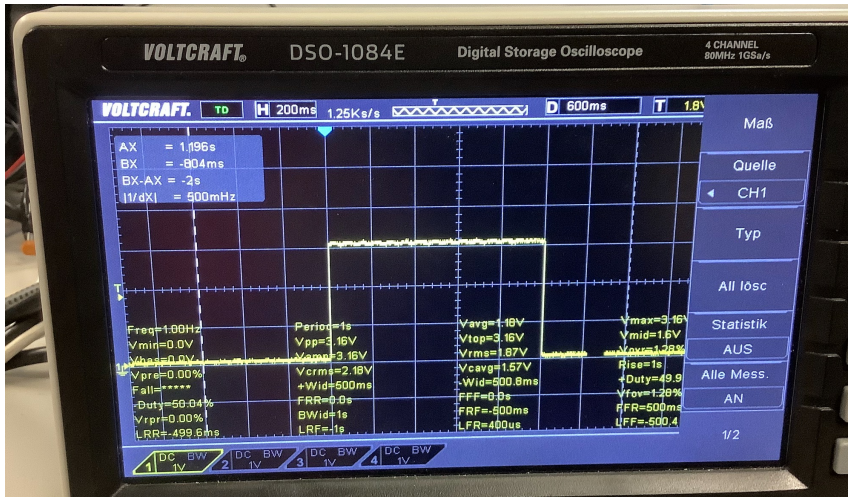
sets Timer/Counter1 of Control Register B
Prescaler to CK/128
Timer1 – 8Bit - $2^8 = 256$
Ex. Quarz = 2^{15} Hz

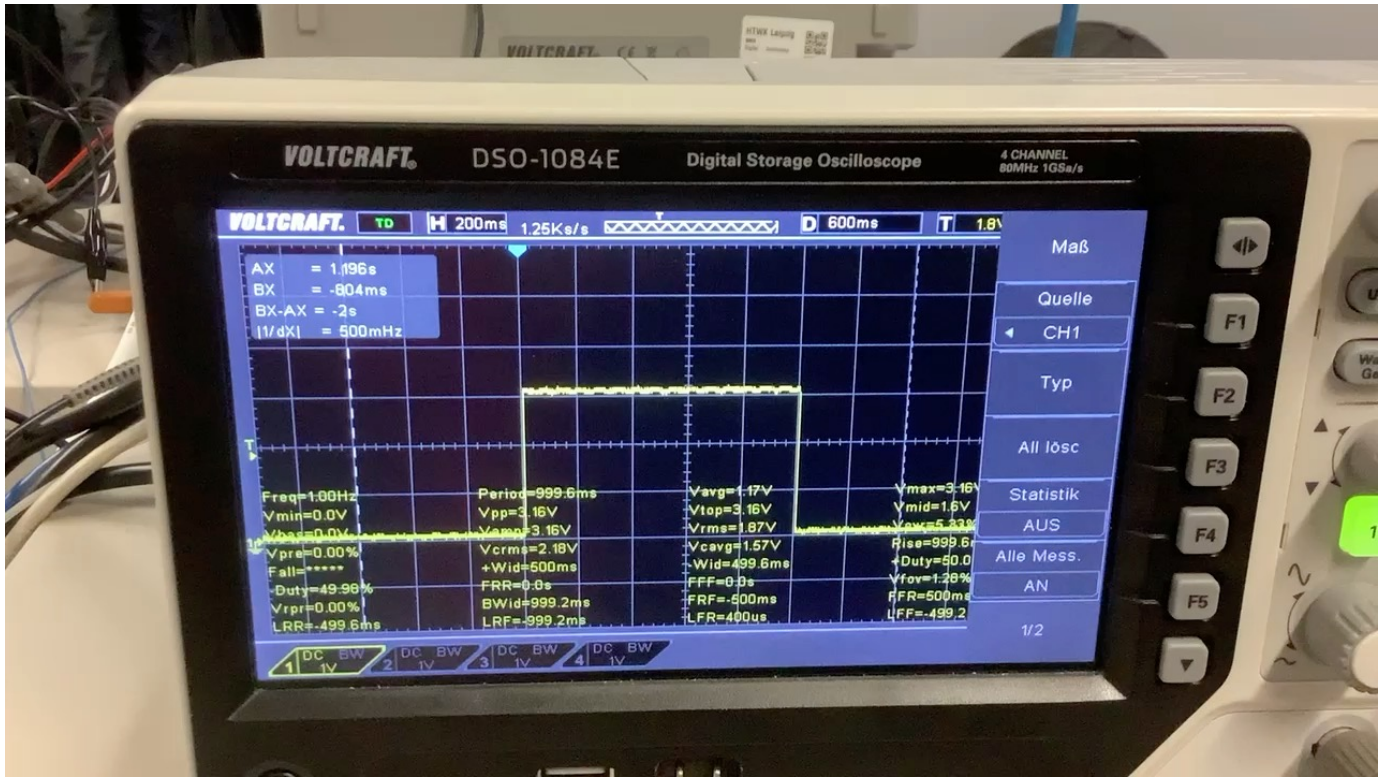
$$Periode = \frac{Prescaler * TimerWidth}{Eingangsfrequenz} = \frac{128 * 2^8}{2^{15}} = 1$$

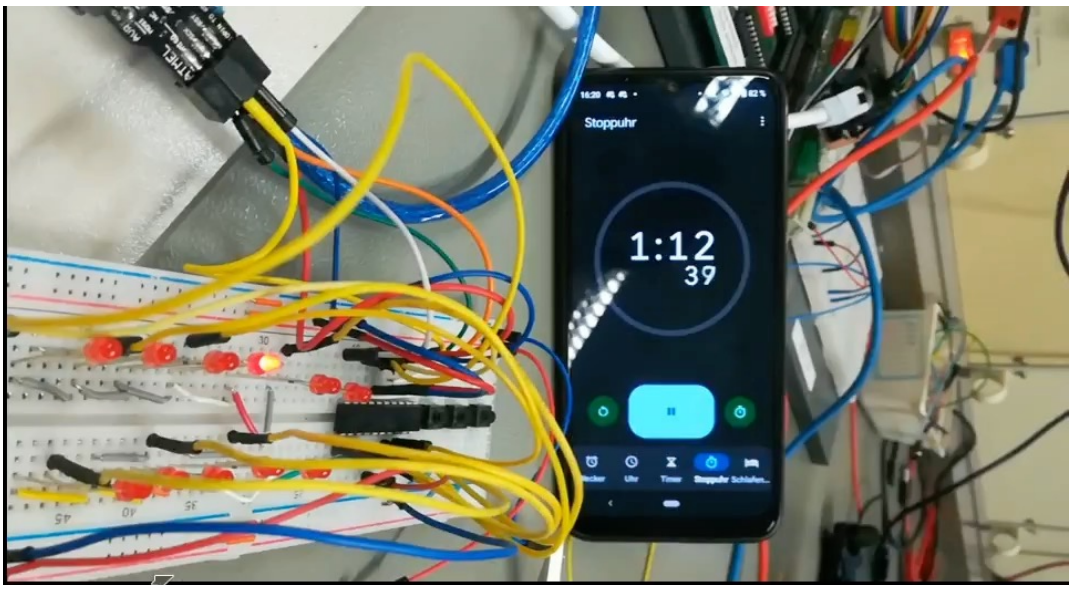
```
ISR(TIMER1_OVF_vect)
{
  seconds++;
  timeSetter();
  sleep++;
}
```

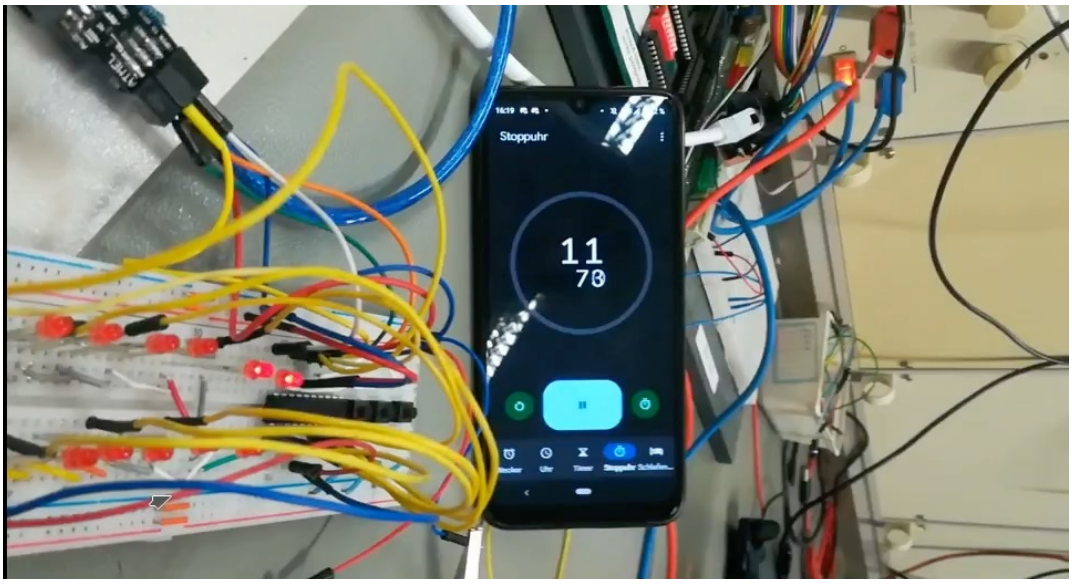
Timer Accuracy

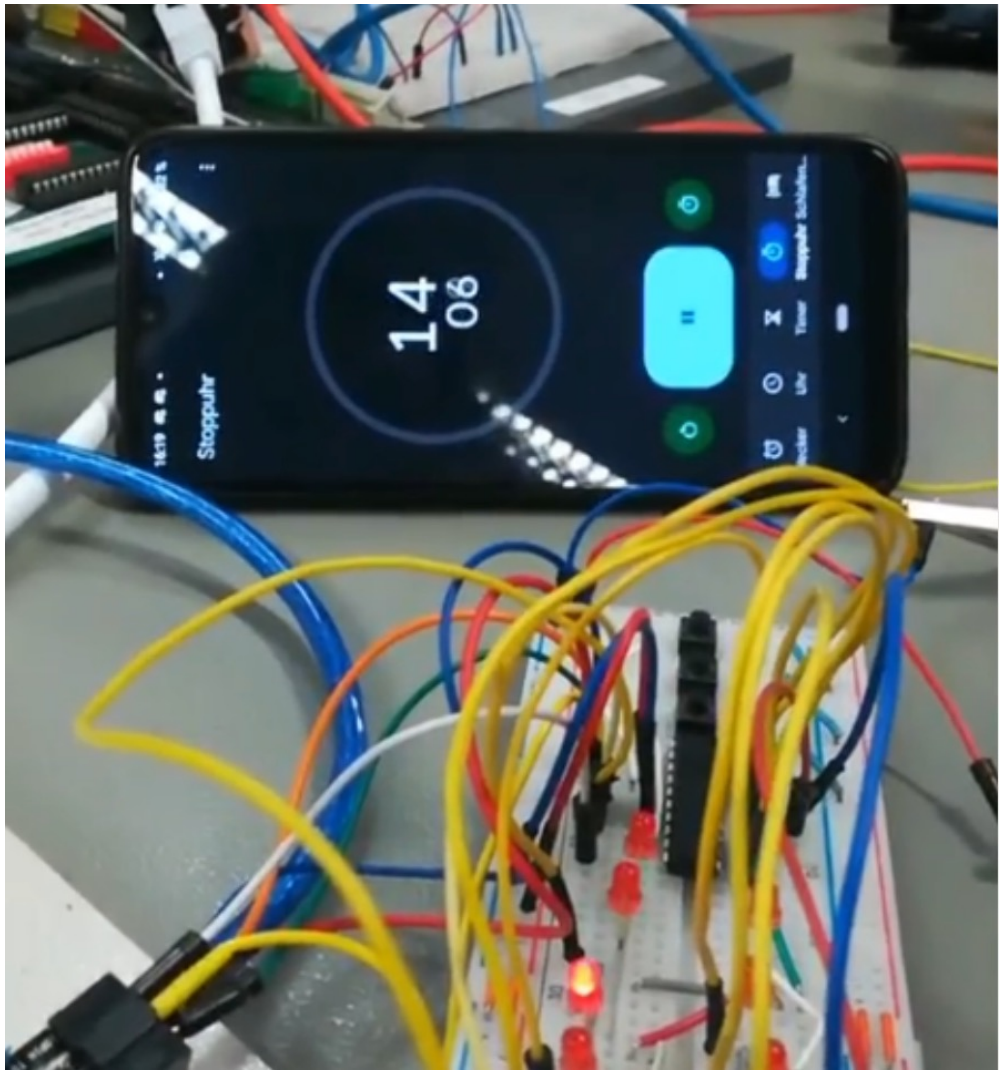
Via Oscilloscope

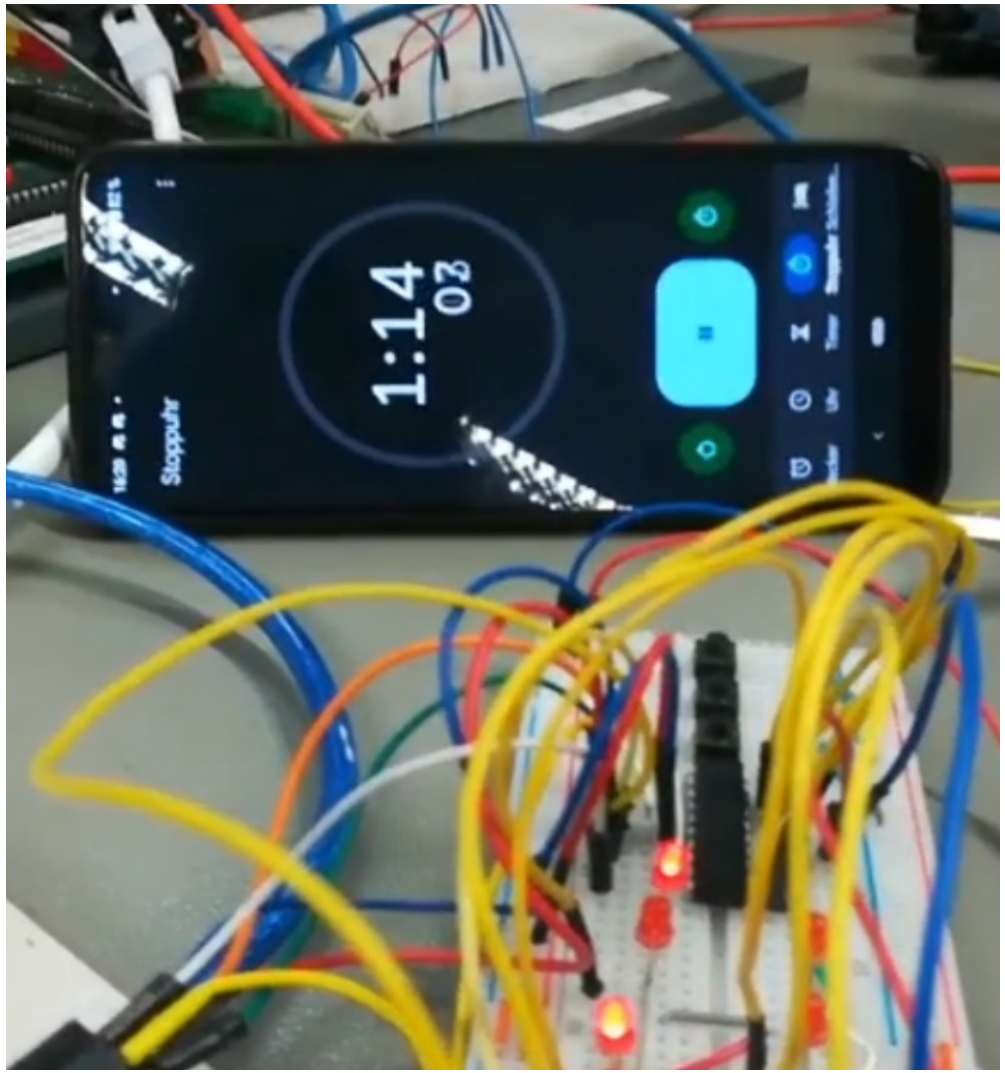





Stoppuhr
1:12
39


Stoppuhr
11
70


14
08
Stoppuhr


1:14
03
Stoppuhr

Set Time

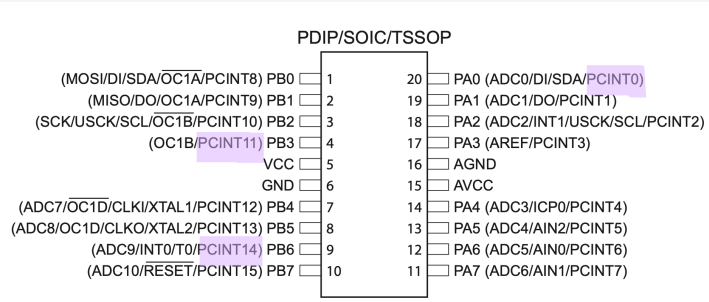
Change global variables

```
/*
 * when seconds are == 60, update LEDSeq && increment hours, minutes
 * when minutes == 60 & hours == 24, reset
*/

void timeSetter()
{
    if (seconds == 60){
        seconds = 0;
        minutes++;
        updateSeq();
    }

    if(minutes == 60) {
        minutes = 0;
        hours++;
    }

    if(hours >= 24) {
        hours = 0;
    }
}
```

Setup

Switches & SleepMode

```
void setup()
{
    DDRA = 0b11111110; // Switch Reg A to 1-7 output and 0 to input
    DDRB = 0b00000000; // Switch Reg B to INPUT (Pin PB6 & PB3 are Interrupts)

    TCCR1B |= (1 << CS13); //set Timer1(Time) auf f/128 bei f= 2^15 overflow nach 1s
    TIMSK |= (1 << TOIE1); // set interrupts for Timer1

    //Pin Change Interrupts - enable PCINT0, PCINT11 und PCINT14
    //Pull up Widerstände setzen
    PORTA |= (1 << PORTA0);
    PORTB |= (1 << PORTB6) | (1 << PORTB3);

    GIMSK |= (1 << PCIE1) | (1<< PCIE0); //enable General Interrupt Mask Register for PCINT[7:0] or PCINT[15:12]
    PCMSK0 |= tastHours; //enable PCINT0 on PORT A PIN A0
    PCMSK1 |= tastMinSleep; //enable PCINT14 und PCINT11 on PORTB PIN B6

    sei(); // enable global interrupts same as SREG |= 0x80;

    /*
        remains enabled: timers, pins pullups, interrupts, Clock, OSC
        disables: CLK_cpu, CLK_flash aka no new instructions are pulled prom flash nor are they processed
    */
    set_sleep_mode(SLEEP_MODE_IDLE);

    //inits LEDs according to initiated global minutes and hours
    updateSeq();
}
```



Pin Change Interrupt Enable

PCIE1: PCINT[0:7] & PCINT[15:12]

PCIE0: PCINT[11:8]

Pin Change Mask Register

Enable PC Int on corresponding I/O Pin

```
#define tastHours 0b00000001;
#define tastMinSleep 0b01001000;
```

Power Management

Leistung

- Ausgehend von einer 3V Batterie

Stromstärke I - LED AN (mA)	Stromstärke I – SLEEP (mA)
21,87	14,07
21,9	14,10
21,88	14,11
21,22	14,02
⌀ 21,51	⌀ 14,05

NORMALBETRIEB

$$P = 3V * 21{,}31 * 10^{-3}\, A = 0{,}06453\, W$$

IDLE

$$P = 3V * 14{,}05 * 10^{-3} = 0{,}04212\, W$$

Leistungsersparnis

$$100\% - \frac{0{,}04212}{0{,}06453} * 100 = 34{,}728\%$$

Interrupts

Pin Change Interrupt for Switches

```
//****Interrupt Serviceroutines**************************************************

ISR(PCINT_vect)
{
    //Debouncing with bouncer variable (bouncer variable incremented in main)
    // determines if button Pin has PINCHANGE to LOW, when true execute code (increment hours, minutes...)

    //PinA0 Interrupt - increment hours, reset bouncer and update(led)Seq
    if (!(PINA & (1<<PINA0))) {
        if(bouncer >= 10) {
            hours++;
            bouncer = 0;
            updateSeq();
        }
    }
    //PinB6 Interrupt - increment minutes, reset bouncer and update(led)Seq
    if (!(PINB & (1<<PINB6))) {
        if(bouncer >= 10) {
            minutes++;
            bouncer = 0;
            updateSeq();
        }
    }

    //PinB3 Interrupt - Wake, sleep reset - sleepmode activated when sleep >= 25 secs
    if (!(PINB & (1<<PINB3))) {

        if(bouncer >= 10) {
            sleep = 0;
            bouncer = 0;
        }
    }

}
```

Update LED Sequence

Adding LED Pin Configurations for showing time to Array

```
/**
 * @brief gets the time and inputs it into an array
*/
void updateSeq()
{
    // set first addr through last adrr of ledSeq to zero
    memset(ledSeq, 0, sizeof ledSeq);

    // disrespect actual time, in case actual time is 00:00
    uint16_t time = 0b0;

    /**
     * set bits 0 - 5 as actual minutes
     * set bits 6 - 10 as actual hours
     */
    time = (hours << 6) ^ minutes;
```

hours = 3 = (0b0000 0011 << 6) = 0b 0000 0000 1100 0000
minutes = 16= XOR 0b 0001 0000

time = 0b 0000 0000 1101 0000

```
// fill array, if time is other than 00:00
if(time > 0b0)
{
    // special index for ledSeq to leave no cell blank
    int arrayInd = 0;

    // iterate 11 times over ledSeq
    for (int i = 0; i < 11; i++)
    {
        /*
        * 0010 (1 will be shifted at each digit position iteration)
        *&1010 (00:10)
        *_____
        * 0010 -> corresponding pin-config will be inserted into ledSeq
        *
        * if result is 0b0, arrayIndex will not be incremented and no pin-config will be inserted
        */
        //if 010(pattern) & 110(current) = 1, then add ledPins of struct ledPairs to ledSeq
        if ((ledPairs[i].bin & time) != 0b0)
        {
            ledSeq[arrayInd] = ledPairs[i].ledPins;
            arrayInd++;
        }
    }
}
```

time = 0b0000 000011010000
& {0b00000001000, 0b01111011}, //D3
bin ledPins

Main

Sleepmodus & Pseudo-PWM

```
int main()
{
    setup();

    while(true)
    {
        if(sleep >= 25) {
            bouncer = 10;
            PORTA = 0x01;
            sleep_mode();
        }
        else {
            bouncer++;
            int i = 0;
            while(ledSeq[i] != 0b0)
            {
                PORTA = ledSeq[i];
                i++;
            }
            i = 0;
        }
    }
}
```

```
* 1. if: check for Sleep
* check if sleep is >= 25 seconds
* set bouncer to 10 to enable wake button
* set all ledPins to zero, so last array value is not on when going to sleep
* put controller to sleep


* 2. else: show Time Part (Pseudo PWM)
* increment bouncer every time main is executed
* go through ledSeq Array which is set in updateSeq() until index is not zero
* show ledSeq on PORTA, when array at index ... is zero, reset index, 2. while condition false
```

Quellen

ATtiny461A
https://ww1.microchip.com/downloads/en/DeviceDoc/doc8197.pdf