

quine/dnf/ttl/co kg/3 in var, gal plus following
atmega8

david vajda

22. Juli 2026

1 quine mc cluskey - 3 in var x_2, x_1, x_0

(C) David Vajda

Mon Jul 20 09:19:18 2026

3 Network - TTL - Disjunktive Normalform

```
x2 x1 x0 y
0 0 0 0 1
1 0 0 1 0
2 0 1 0 0
3 0 1 1 0
4 1 0 0 0
5 1 0 1 1
6 1 1 0 0
7 1 1 1 1
```

```
    x2 x1 x0 y
0 0 0 0 1
5 1 0 1 1
7 1 1 1 1
```

```
    x2 x1 x0 y
Gruppe 0:
0 0 0 0 1
Gruppe 2:
5 1 0 1 1
Gruppe 3:
7 1 1 1 1
```

```
x2 x1 x0 y
Gruppe 0:
0 0 0 0 1
5:7 1 -1 1
```


Y = /X2 & /X1 & /X0 # X2 & X0

David Vayda (c)

07/20/2026

Seite 1, duf - quine(4)3

jetzt...

da (die) das selbstgeschriebene programm
auf einmal einen seltsamen Fehler
zeigt... den er trotz 4 Variablen für
das es geschrieben ist - und jetzt 3
benutzt wurden - Eingangsvariablen -
wo sicher davon aus zu gehen ist -
dass es keine Rolle spielt - eben.
wg. quine - und: das Programm sehr
klein ist mühsam selbst geschrieben
in Python 3 - macht die Testphase
wahnsinnig. Deswg. mache ich dasselbe
per Hand. Ich kann die Formel auch
per Hand überprüfen. Und ebenso per
Latex ins Timing Diagramm eintragen.
Dass sie richtig ist möchte ich eben
zeigen. Das kann ich per Hand machen.
Zeigt sich nachher im gal16v8...
mit galasm.
nächste Seite: Die Tabelle + Handgezeichnetes
Diagramm.

David Vajda (c)

07/20/2026

Seite 3, Versuch, quine KVL... chuf..

neuartiges wahrscheinlich altes verfahren
Tabellen zusammen, indem alle 0en
ausgeschaltet werden indem 1er in
y stehen

Gleichung für y ist oder?

natürlich: chuf = min und
Maxterm

Tabelle 0, Maxterm M_0

x_2	x_1	x_0	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Tabelle 1

x_2	x_1	x_0	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

x_2	x_1	x_0	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

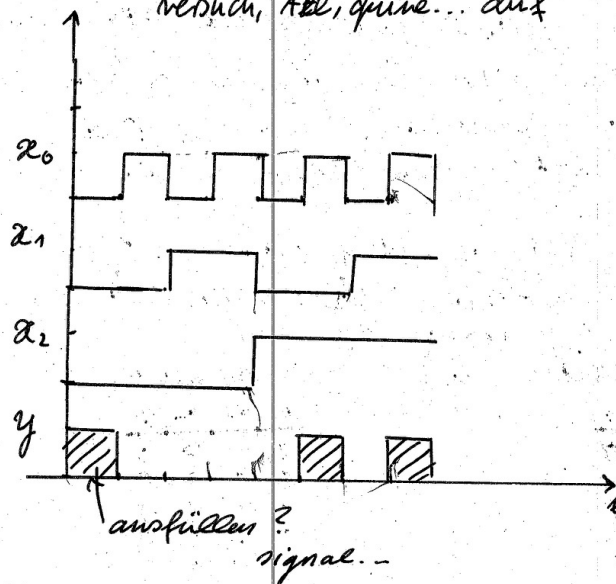
ok...

Dann kommt das timing diagram

David Vajda (c)

07/20/2026, Seite 4

versuch, Abl, quine... darf



ich denke die formel bestätigt sich

David Nayda (c)

07/19+1=20/2028

Seite 2, versch, quine etc... auf...

formel:

$$y = (\neg x_2 \wedge \neg x_1 \wedge \neg x_0) \vee (x_2 \wedge x_0)$$

erste tabelle...

falsch: zerlegung

min - terme

max - terme!

Achtung neu lernen

↓

$$M_0 = (\neg x_2 \wedge \neg x_1 \wedge \neg x_0)$$

$$M_1 = (x_2 \wedge x_0)$$

1. Tabelle für 1. Maxterm

	x_2	x_1	x_0	y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

kann man sagen wo Bedingung nicht erfüllt ist
Bedingung und gleichung gleiches da

2. Tabelle für 2. Maxterm

	x_2	x_1	x_0	y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

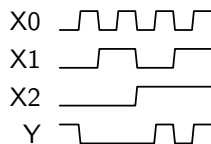
2. links machen wo 1

in folg

x

Verzweigung und bool: spruch Bestimmung

aristoteles und



David Vayda (c)

07/20/2026, Seite 5

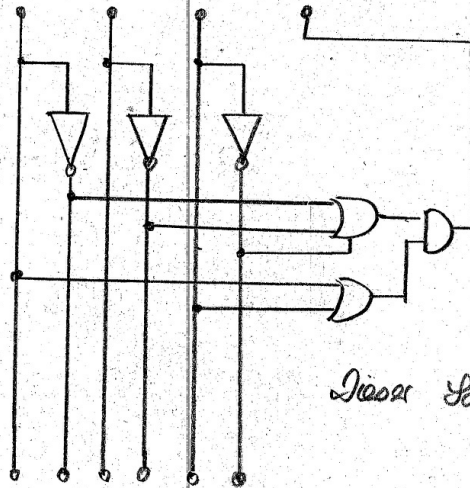
duf, ..., quine usw Schaltplan...

und / and

or

not / inverter / complement

ieee 91 - 1984



Dieser Schaltplan...

wenn man sich das anguckt



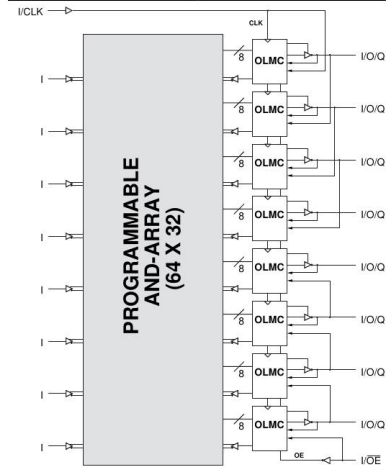
GAL16V8

High Performance E²CMOS PLD
Generic Array Logic™

Features

- **HIGH PERFORMANCE E²CMOS® TECHNOLOGY**
 - 3.5 ns Maximum Propagation Delay
 - F_{max} = 250 MHz
 - 3.0 ns Maximum from Clock Input to Data Output
 - UltraMOS® Advanced CMOS Technology
- **50% to 75% REDUCTION IN POWER FROM BIPOLAR**
 - 75mA Typ I_{cc} on Low Power Device
 - 45mA Typ I_{cc} on Quarter Power Device
- **ACTIVE PULL-UPS ON ALL PINS**
- **E² CELL TECHNOLOGY**
 - Reconfigurable Logic
 - Reprogrammable Cells
 - 100% Tested/100% Yields
 - High Speed Electrical Erasure (<100ms)
 - 20 Year Data Retention
- **EIGHT OUTPUT LOGIC MACROCELLS**
 - Maximum Flexibility for Complex Logic Designs
 - Programmable Output Polarity
 - Also Emulates 20-pin PAL® Devices with Full Function/Fuse Map/Parametric Compatibility
- **PRELOAD AND POWER-ON RESET OF ALL REGISTERS**
 - 100% Functional Testability
- **APPLICATIONS INCLUDE:**
 - DMA Control
 - State Machine Control
 - High Speed Graphics Processing
 - Standard Logic Speed Upgrade
- **ELECTRONIC SIGNATURE FOR IDENTIFICATION**
- **LEAD-FREE PACKAGE OPTIONS**

Functional Block Diagram



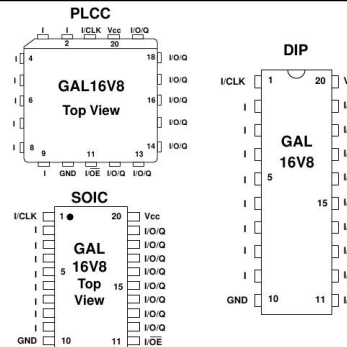
Description

The GAL16V8, at 3.5 ns maximum propagation delay time, combines a high performance CMOS process with Electrically Erasable (E²) floating gate technology to provide the highest speed performance available in the PLD market. High speed erase times (<100ms) allow the devices to be reprogrammed quickly and efficiently.

The generic architecture provides maximum design flexibility by allowing the Output Logic Macrocell (OLMC) to be configured by the user. An important subset of the many architecture configurations possible with the GAL16V8 are the PAL architectures listed in the table of the macrocell description section. GAL16V8 devices are capable of emulating any of these PAL architectures with full function/fuse map/parametric compatibility.

Unique test circuitry and reprogrammable cells allow complete AC, DC, and functional testing during manufacture. As a result, Lattice Semiconductor delivers 100% field programmability and functionality of all GAL products. In addition, 100 erase/write cycles and data retention in excess of 20 years are specified.

Pin Configuration



Copyright © 2004 Lattice Semiconductor Corp. All brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

LATTICE SEMICONDUCTOR CORP., 5555 Northeast Moore Ct., Hillsboro, Oregon 97124, U.S.A.
Tel. (503) 268-8000; 1-800-LATTICE; FAX (503) 268-8556; <http://www.latticesemi.com>

August 2004

16v8_09

1

wenn man sich die pins anguckt...

GND

ist im sued-westen, der pin. dann im nord-osten, ist

VCC

dass die eine reihe, in diesem quasi von west nach ost, der waagerechte vertauscht ist, hat in diesem falle keine auswirkung. wir sehen aber:

```
/OE VCC
CLK GND
```

ist das erfuehlt. ja!

vertauscht sind

```
Y: OUT
X2, X1, X0: IN
```

vielleicht erst die eine reihe, dann die andere.

jetzt ist es richtig, und es haben noch pins, nicht 8 auf jeder seite, sondern 10, war eigentlich klar: aber es tut jetzt!!!!

```
GAL16V8
MODULE GAL16v8QUINE07202026

CLK X2 X1 X0 NC NC NC NC NC GND
/OE Y NC NC NC NC NC NC NC VCC

Y = /X2 & /X1 & /X0 # X2 & X0
```

```
DESCRIPTION
GAL16v8QUINE07202026 ...
```

so, die ausgabe lautet:

```
david@www001:~$ ./galasm gal07202026.gal
GALasm 2.1, Portable GAL Assembler
Copyright (c) 1998-2003 Alessandro Zummo. All Rights Reserved
Original sources Copyright (c) 1991-96 Christian Habermann

Assembler Phase 1 for "gal07202026.gal"
Assembler Phase 2 for "gal07202026.gal"
GAL16V8; Operation mode: simple; Security fuse off
Assembling successfully completed.
david@www001:~$
```

das jetzt programmieren und: das ding einsetzen und dann TTL Gatter nehmen, die schaltung bauen, danach mit dem atmega8, sollte die zeit bis morgen nicht reichen, fuer

HD44780

und serial-wire (rs232 in dem falle), dann: mache ich an der stelle morgen weiter, bis heute nachmittag gelernt wird.

jetzt kommt es durch das programm

galasm

durch das uebersetzen der Quelldatei, zu folgenden Pinbelegung, die werden von dem Programm miterzeugt

GAL16V8

-----___/-----			
CLK		1	20 VCC
X2		2	19 NC
X1		3	18 NC
X0		4	17 NC
NC		5	16 NC
NC		6	15 NC
NC		7	14 NC
NC		8	13 NC
NC		9	12 Y
GND		10	11 /OE

und wir sollten bedenken, auf so einem GAL sind Fuses, einige Dateien enthalten, so zu sagen, Informationen an den User, andere sind dann fuer den Baustein

jedec

sind Bauteilbezeichnungen in den USA - nach dem Standard

IEEE - International Electrical and Electronical Engineers

und: die Typennummer/Bezeichnungen, von Dioden, Transistoren sind

jedec

aber auch Schaltpläne, z.B.

IEEE 91-1984

diese Datei mit

jedec

enthält wiederum den Binary fuer das GAL. mit dem Begriff Typennummer und Type aufpassen, es gibt Begriffe, wie

1. kennziffer, kennnummer

2. nennwert

hier muesste genau zwischen nummer, ziffer, das gibt es in der wirtschaft an verschiedenen stellen, unterscheiden. die type selber, oder der . hier muss man beim kauf des programmiergeraets aufpassen, immer lesen, was da steht, das andere um einen herum, klaert der herr gott.

jedenfalls: ein programmiergeraet, hat eine typenbezeichnung. oder den typ. der typ selber, ist die sache, also die lok, bei der Bahn, oder

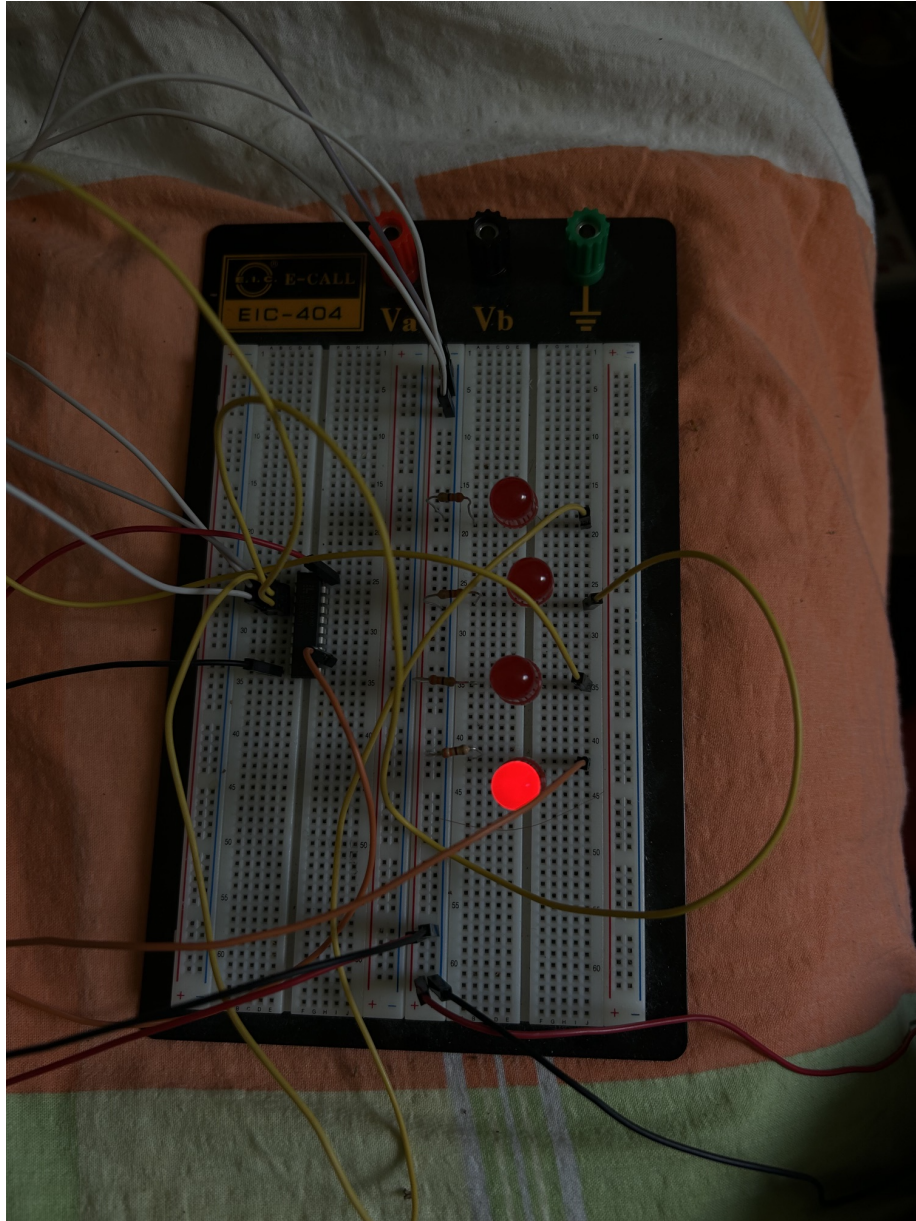
STLINK-V3SET

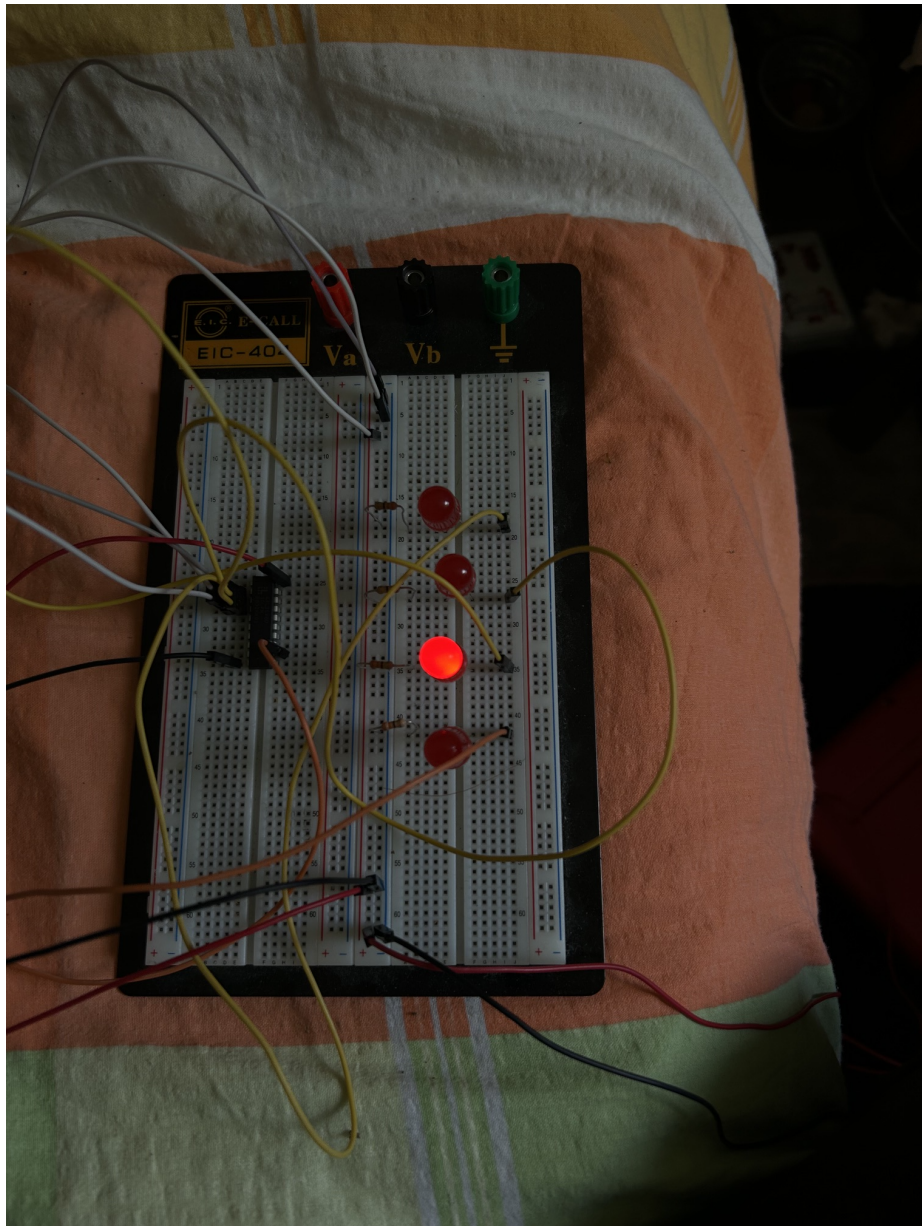
das ist der typ, oder die typenbezeichnung, so wie der hersteller, so wie aber wenn eine Diode, einen typ hat, verschiedene hersteller, dann ist die

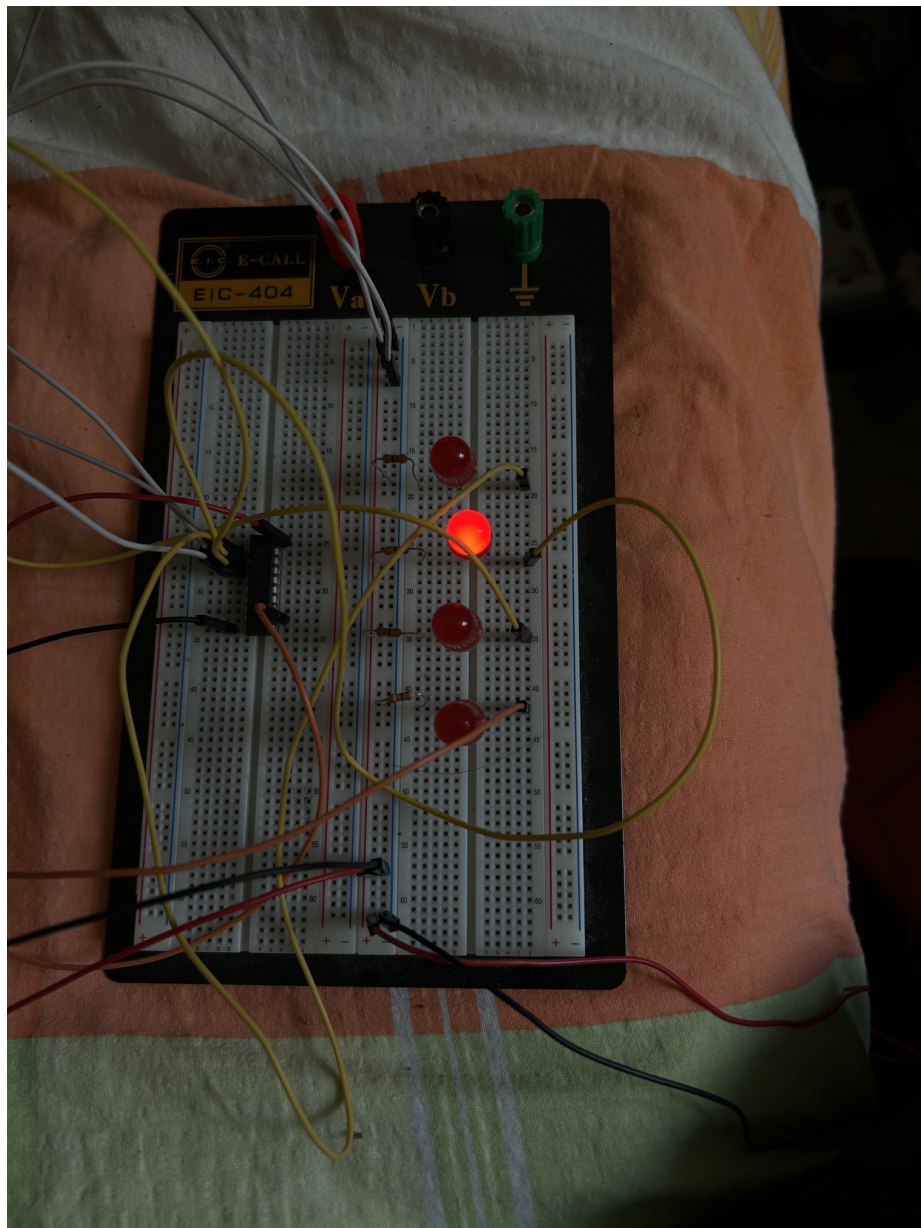
typennummer

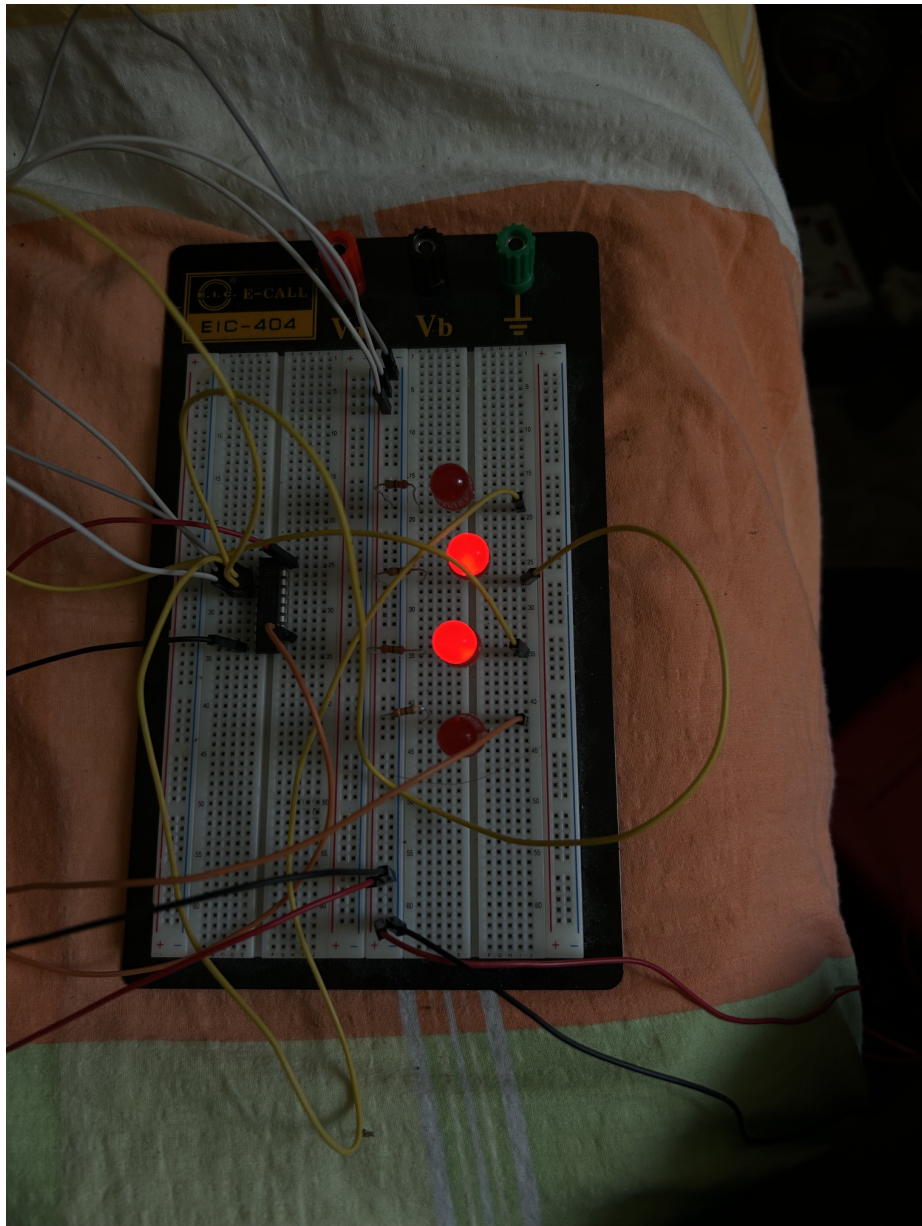
drauf, und: es hat einen typ, aber das kommt von allen herstellern.

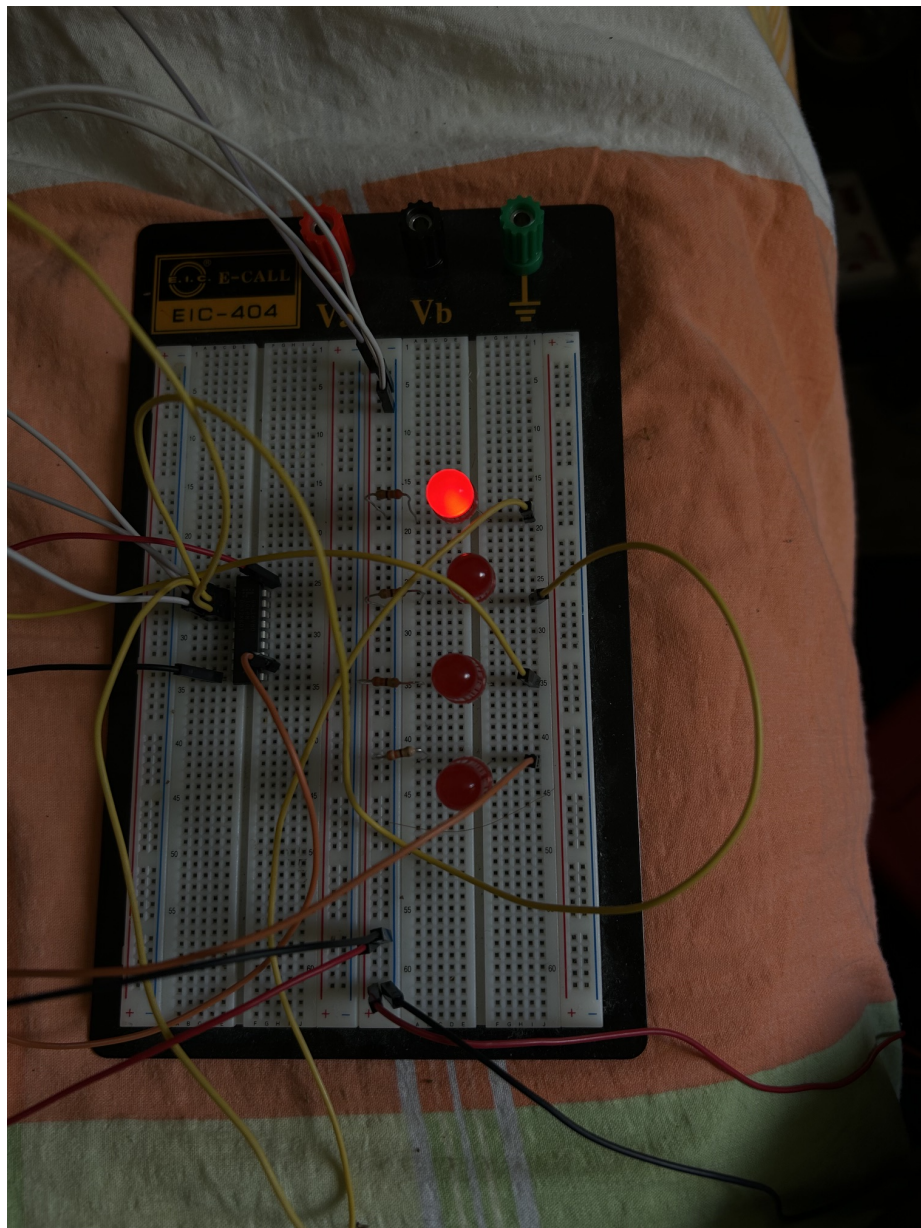
1.2 mit TTL und GAL16v8

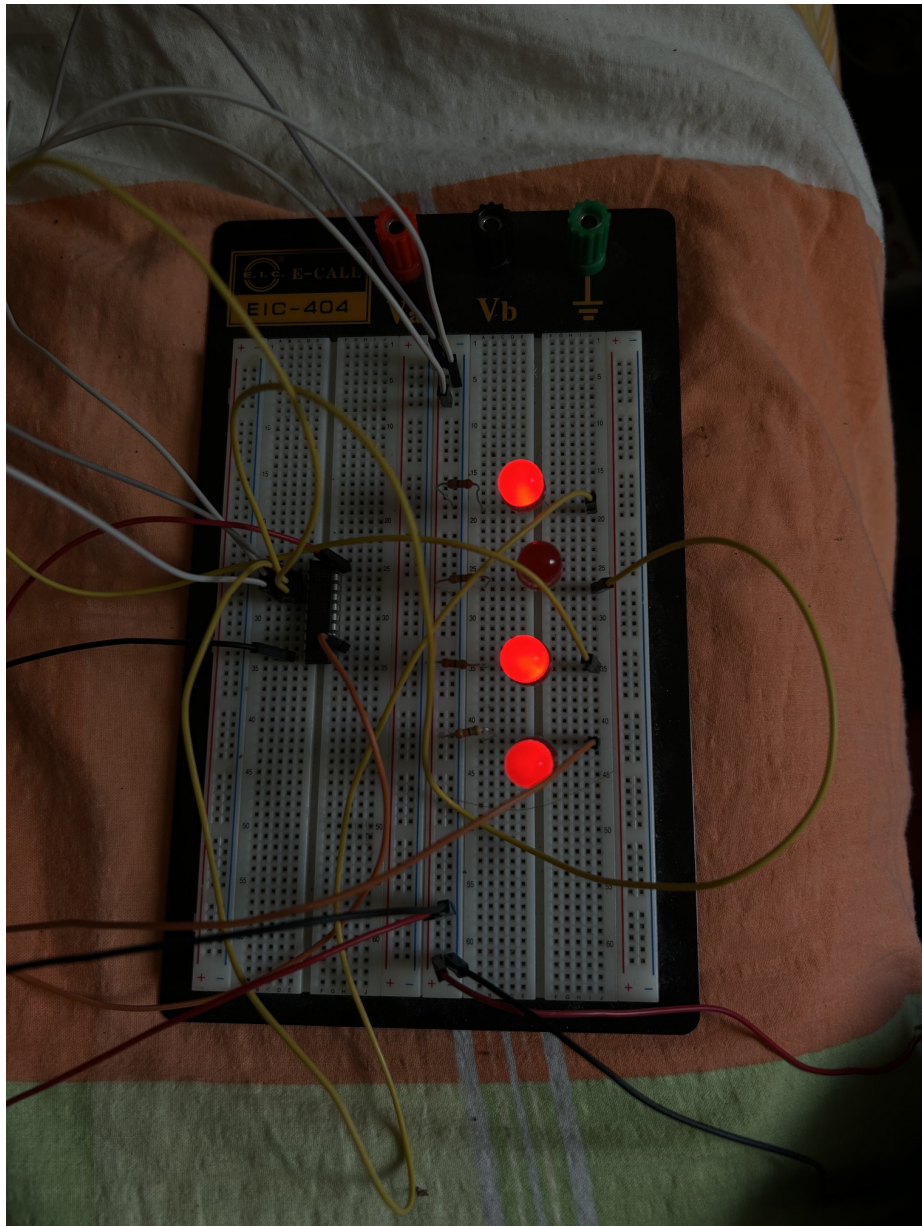


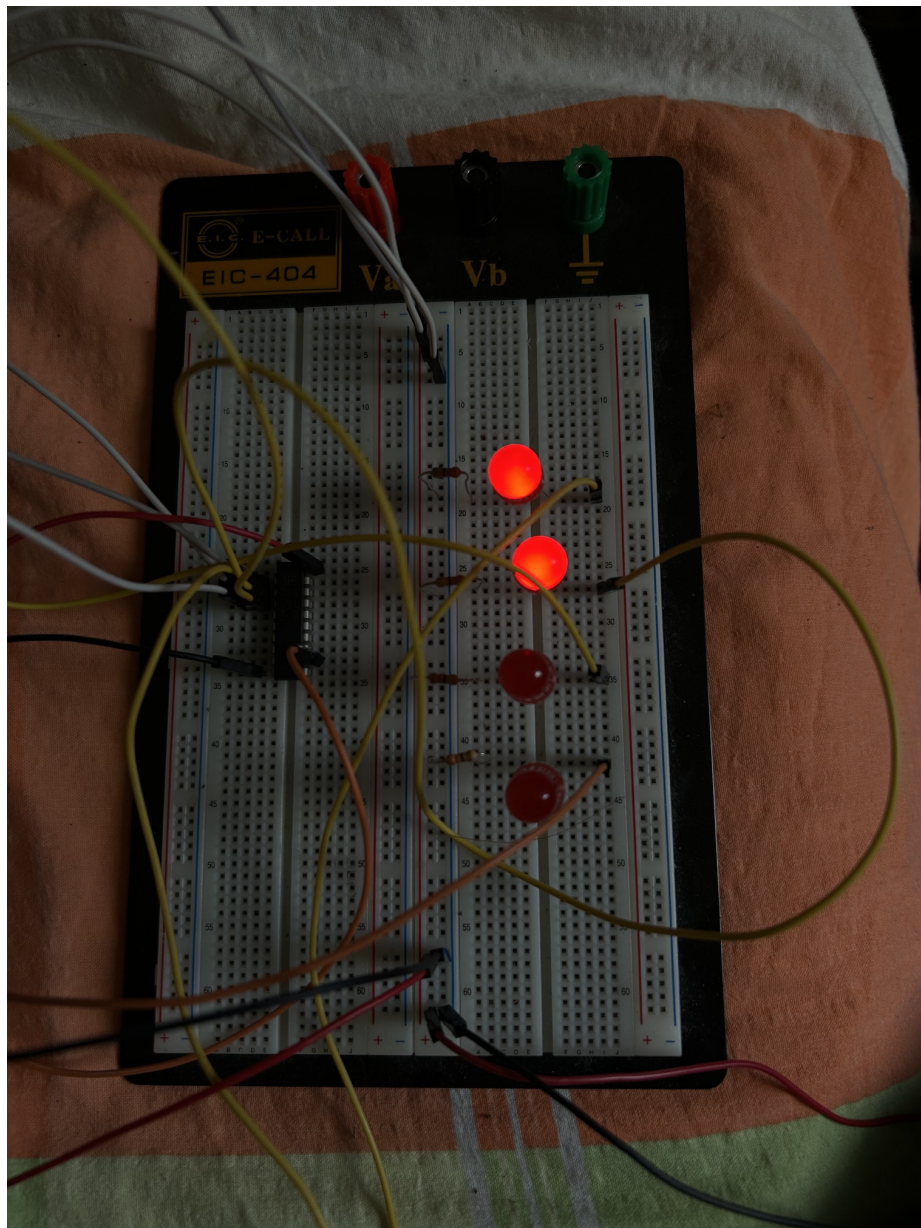


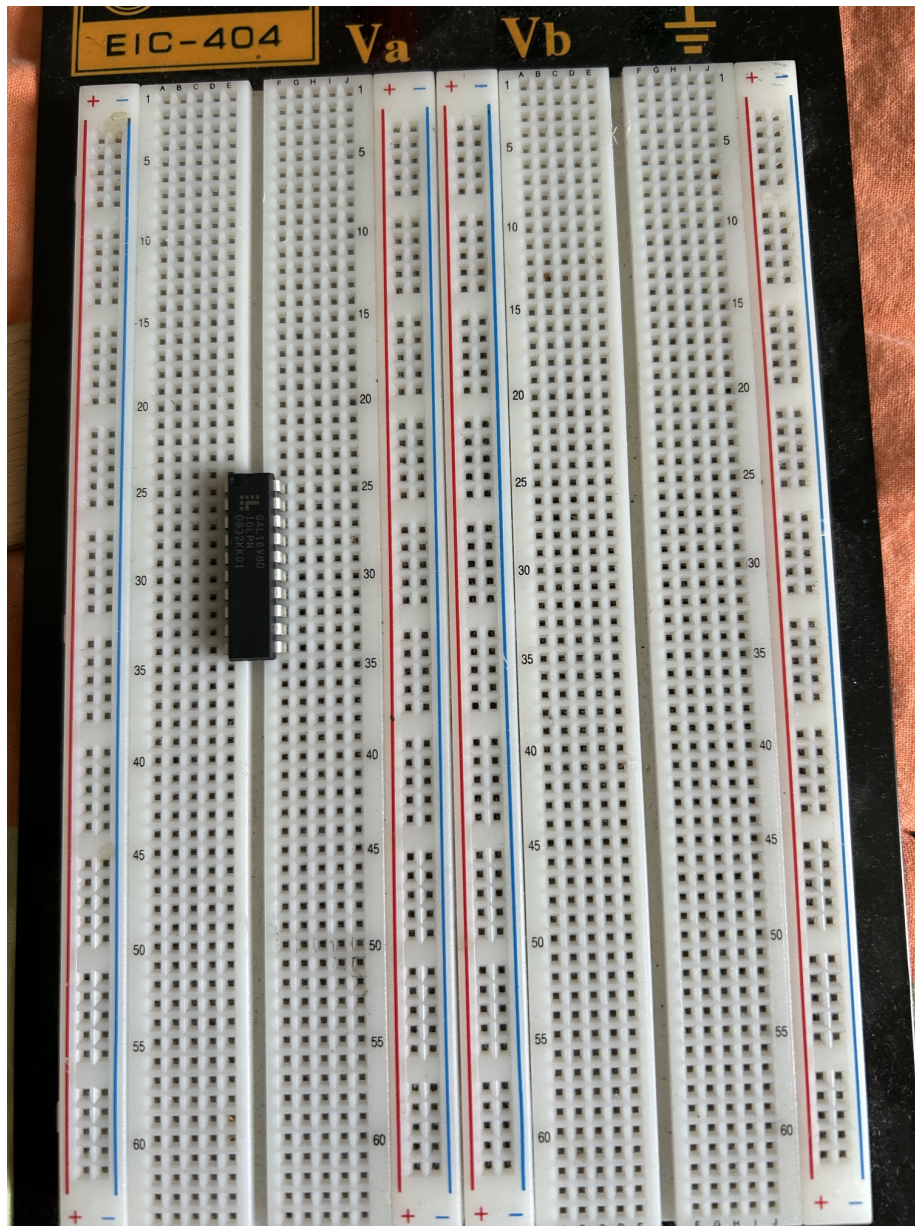


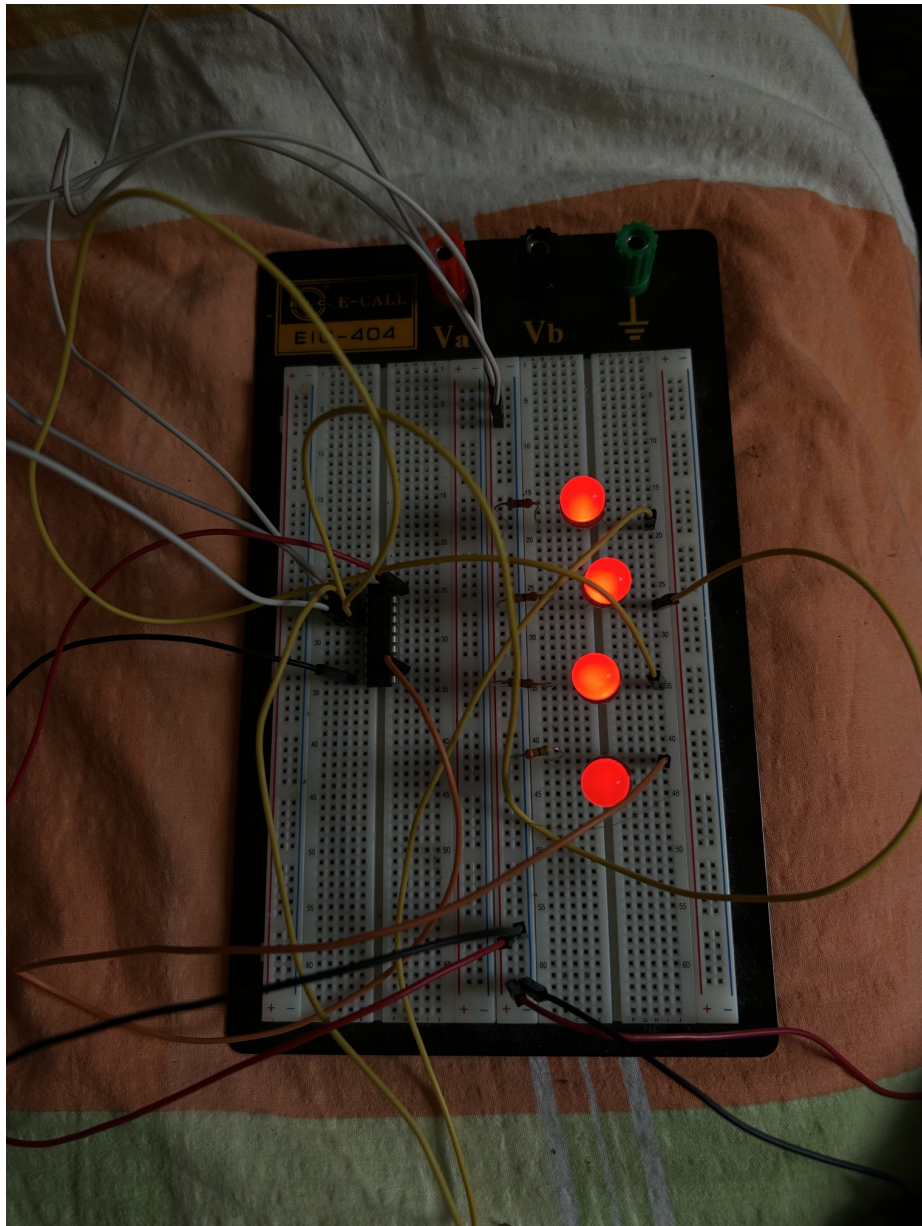


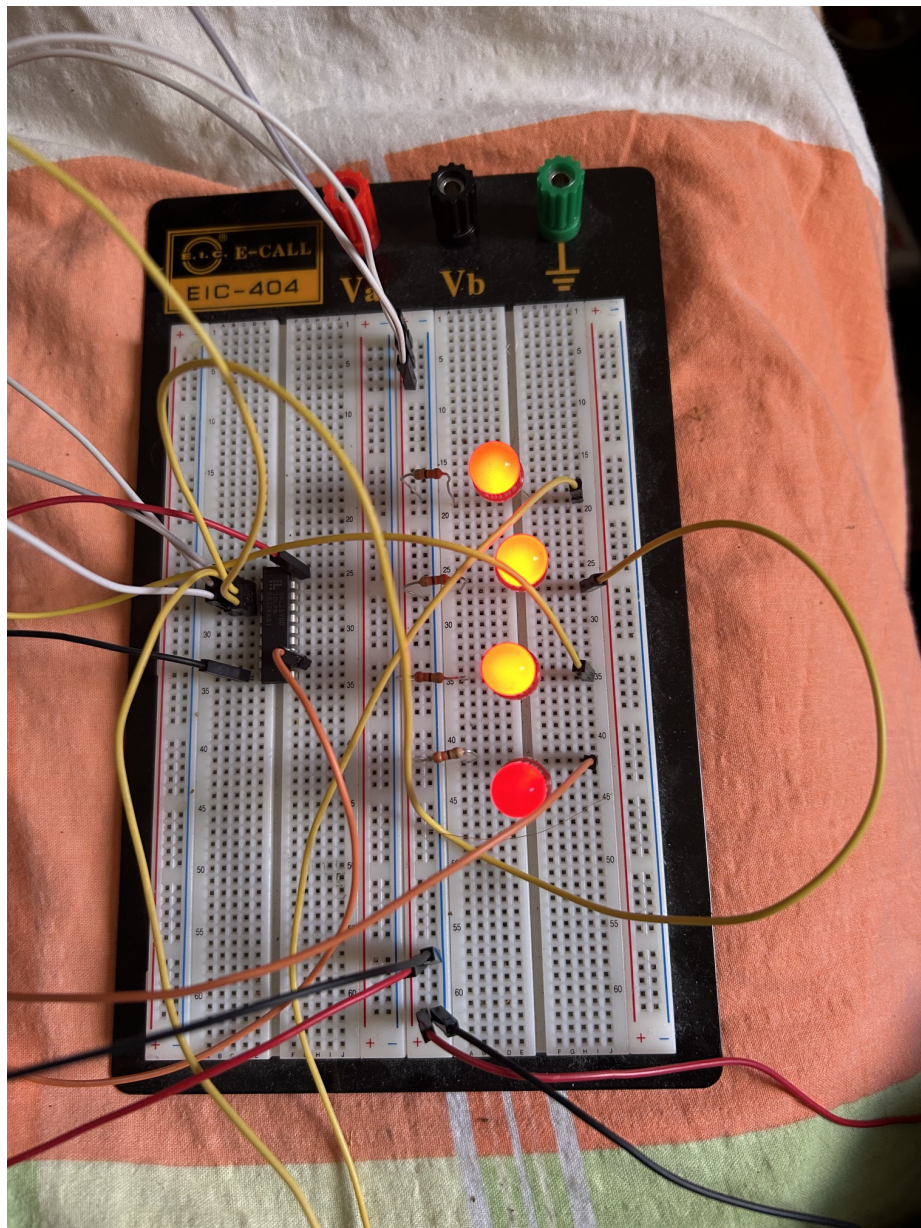


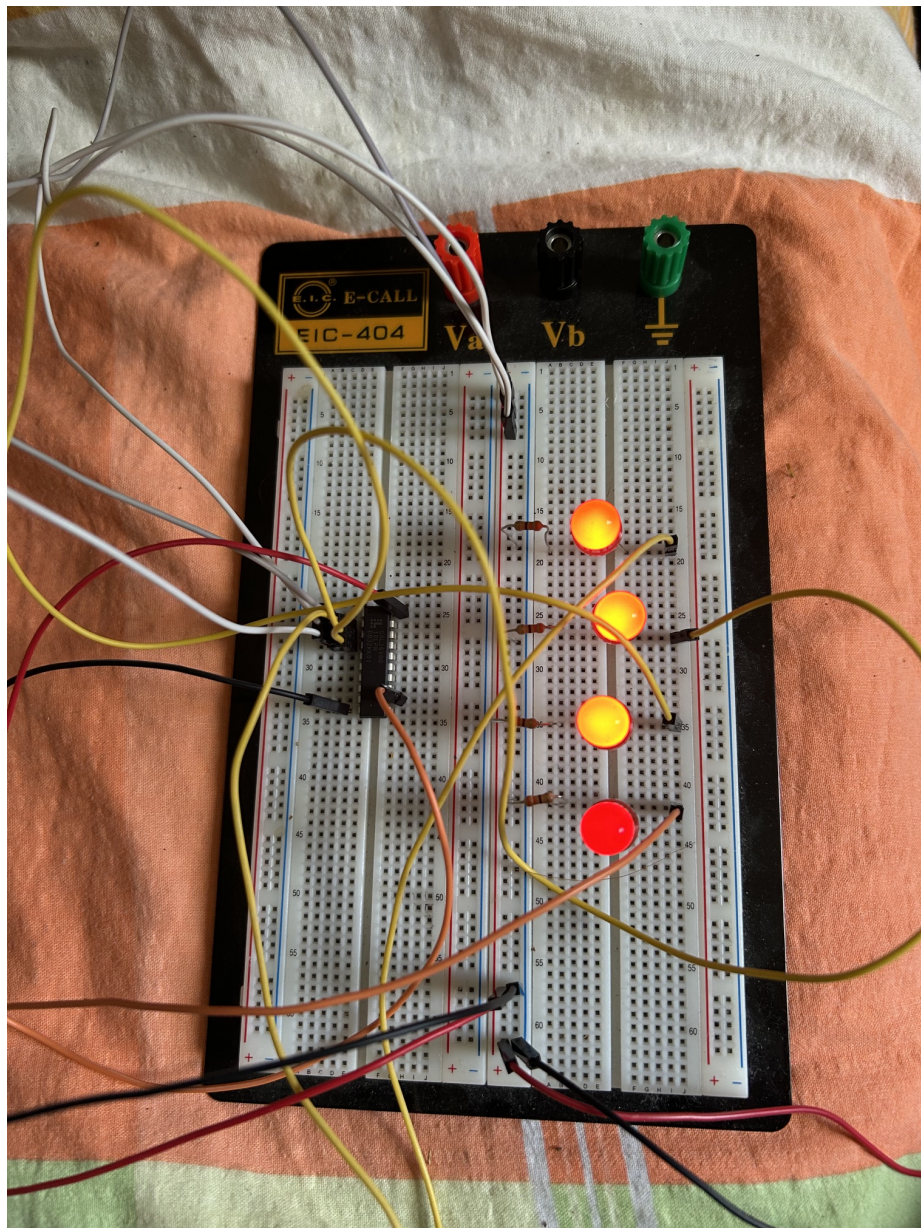


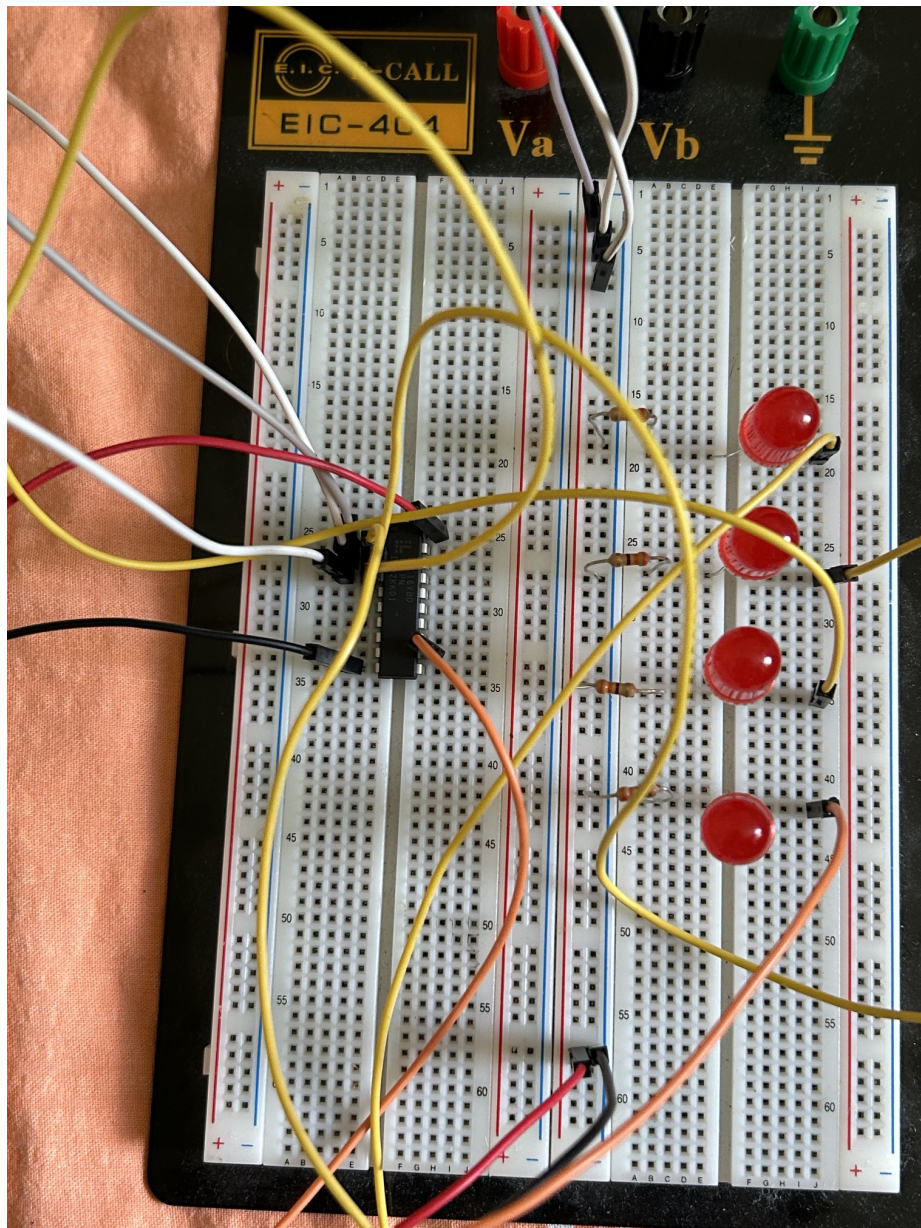


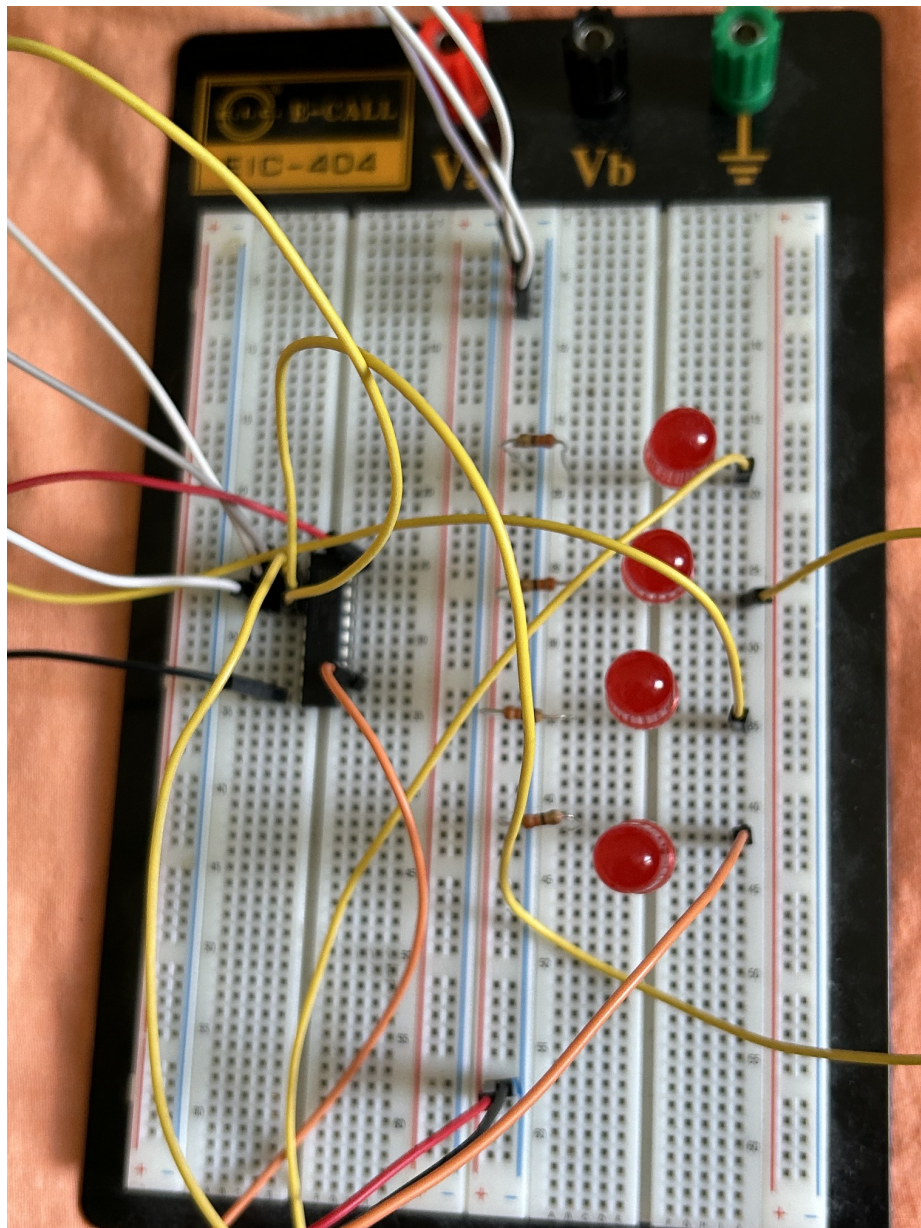


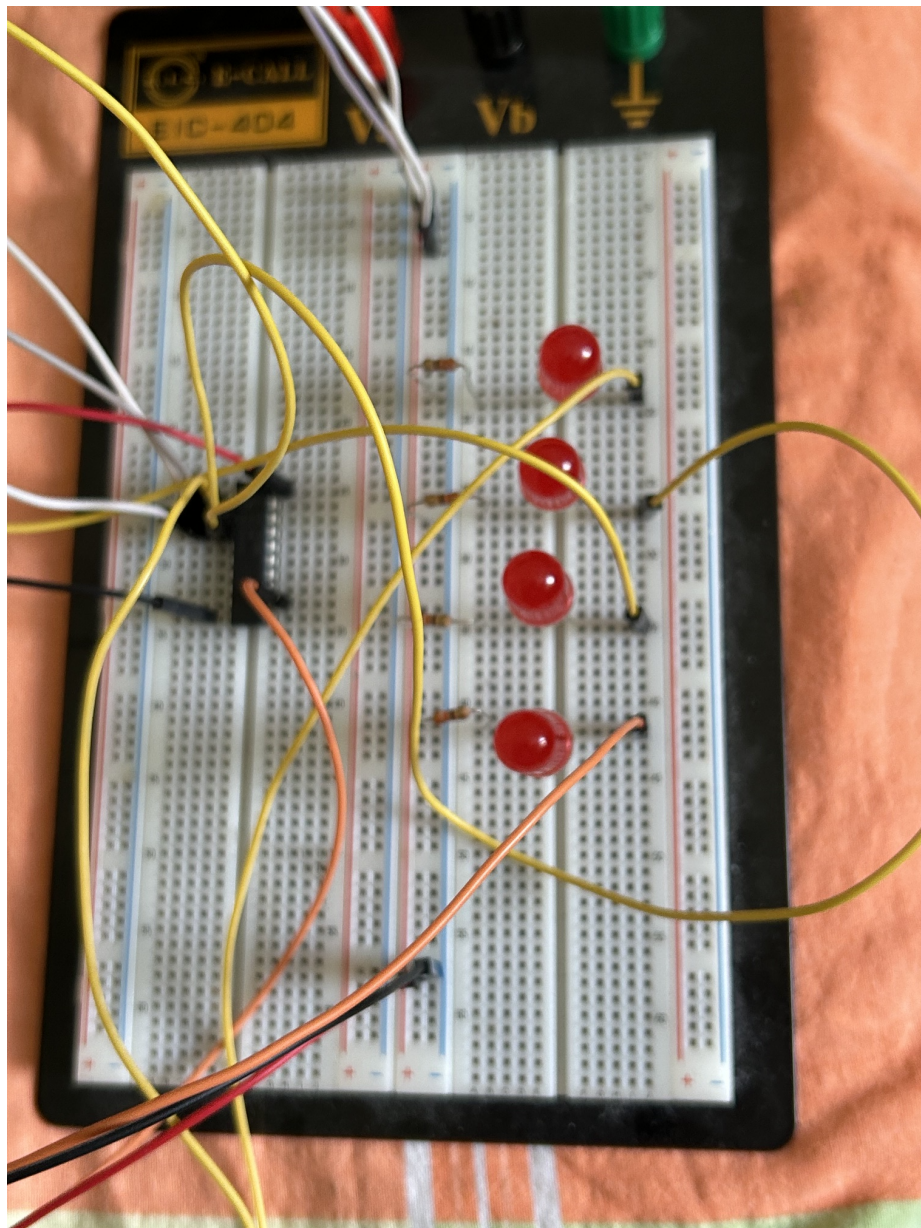


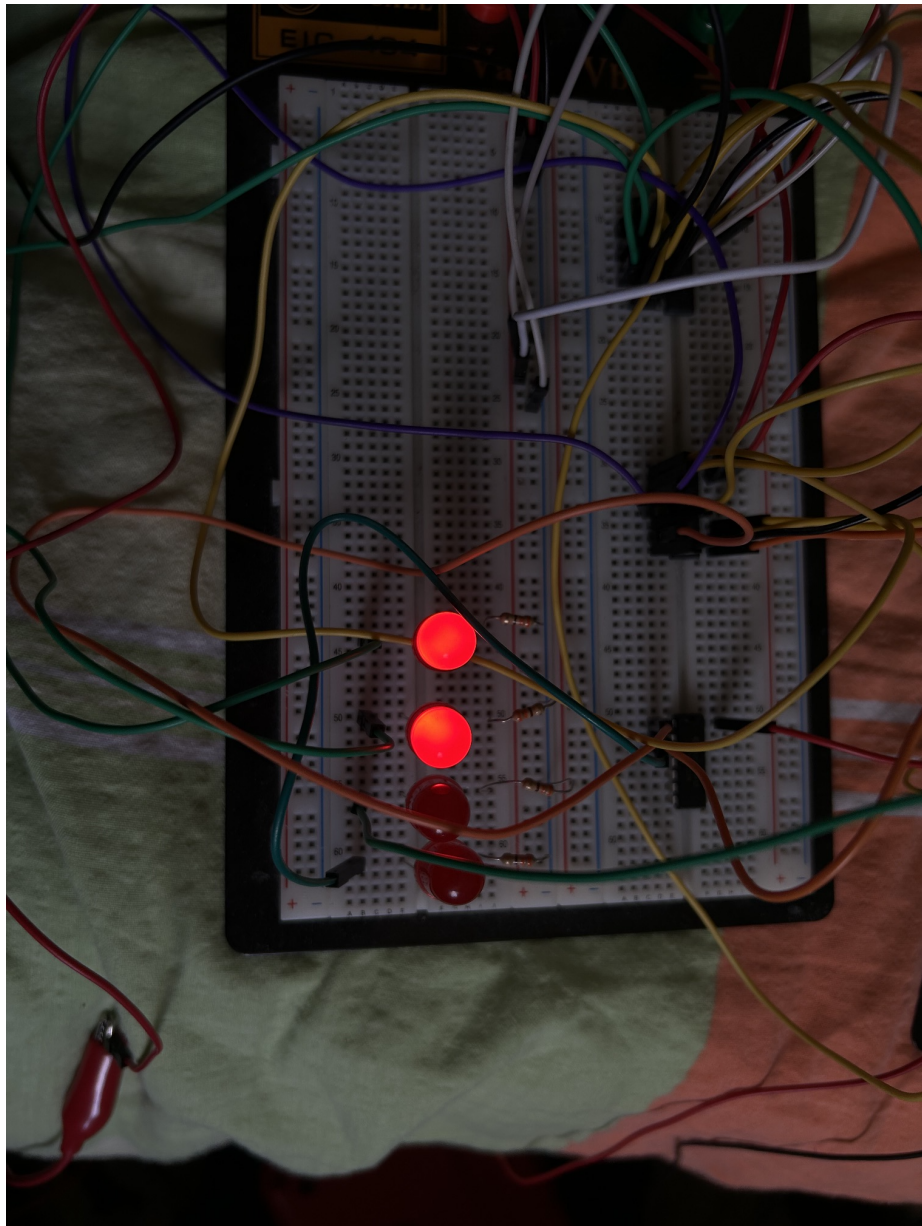


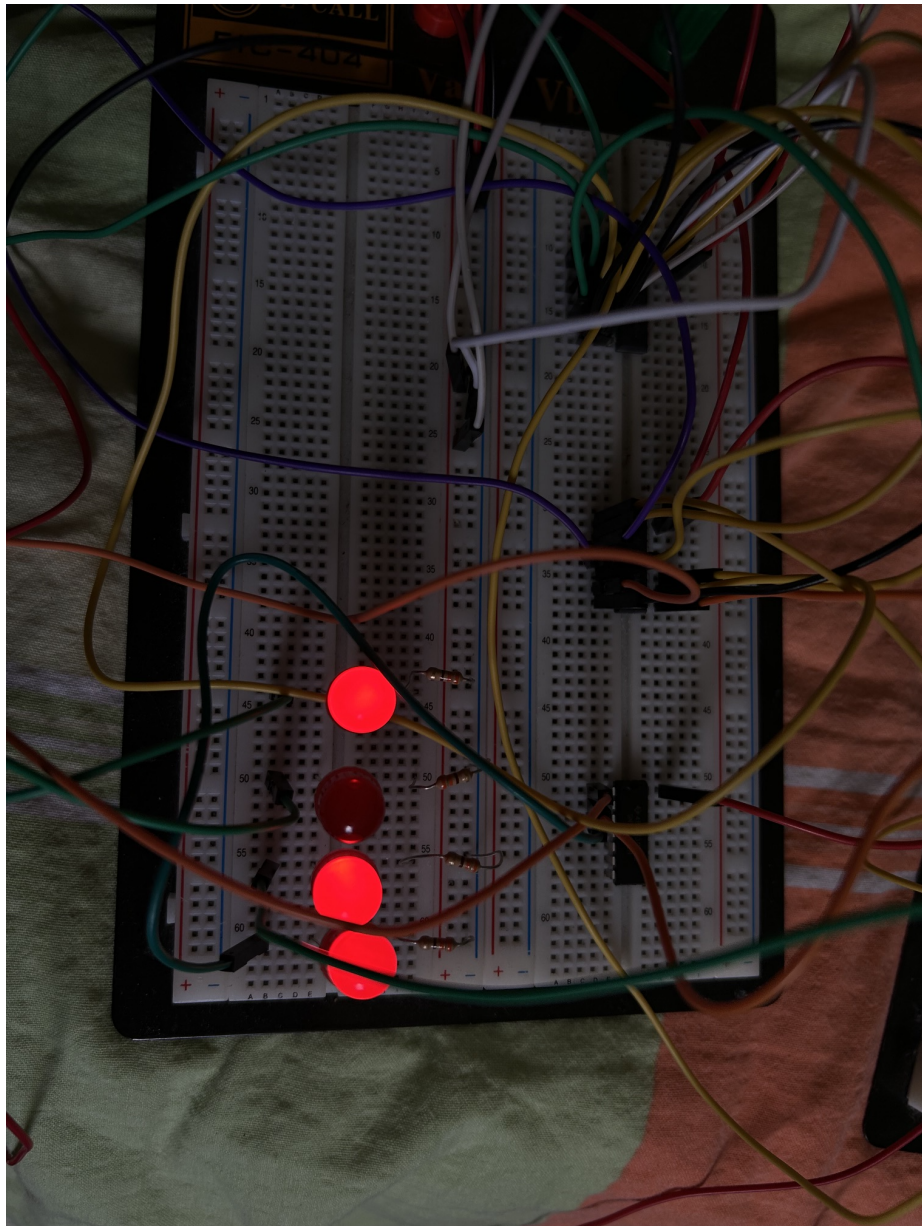


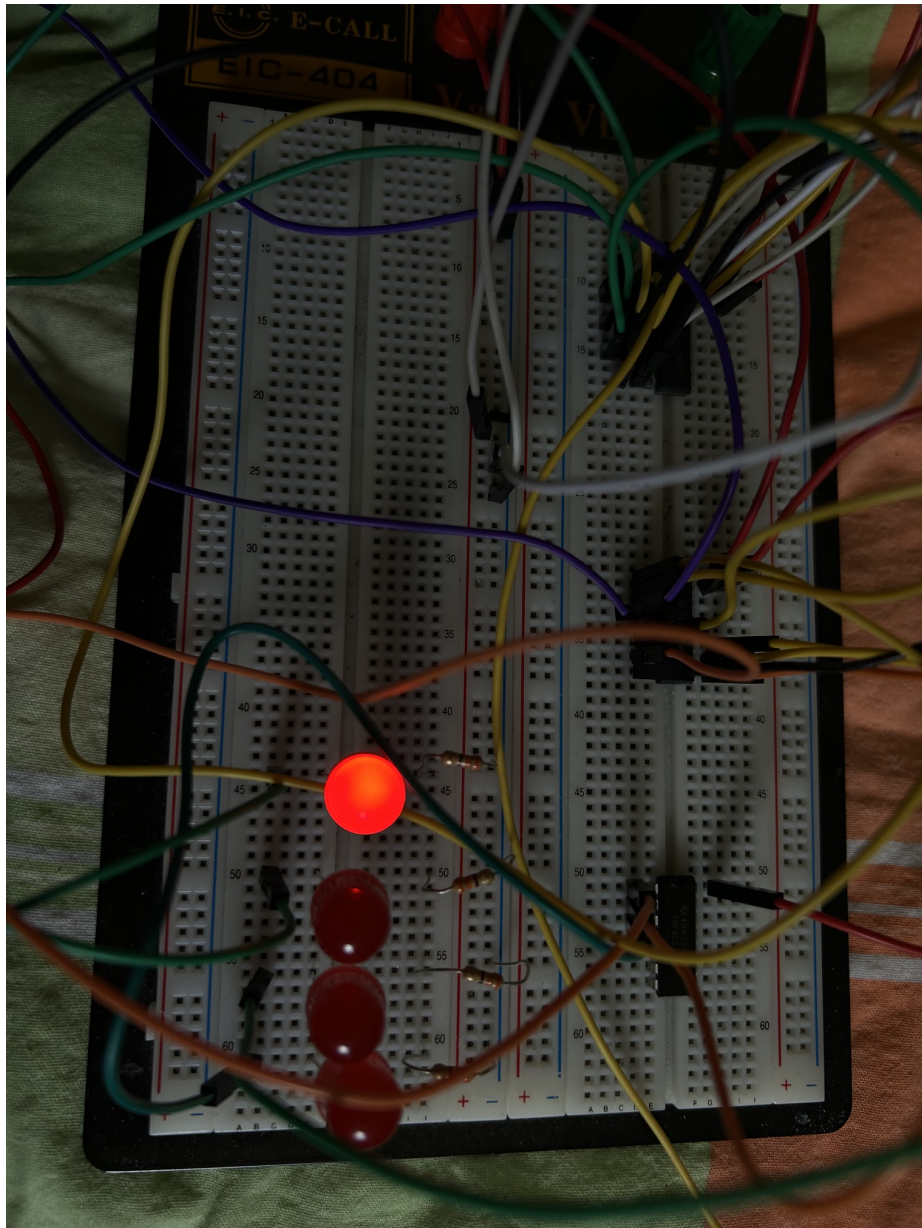


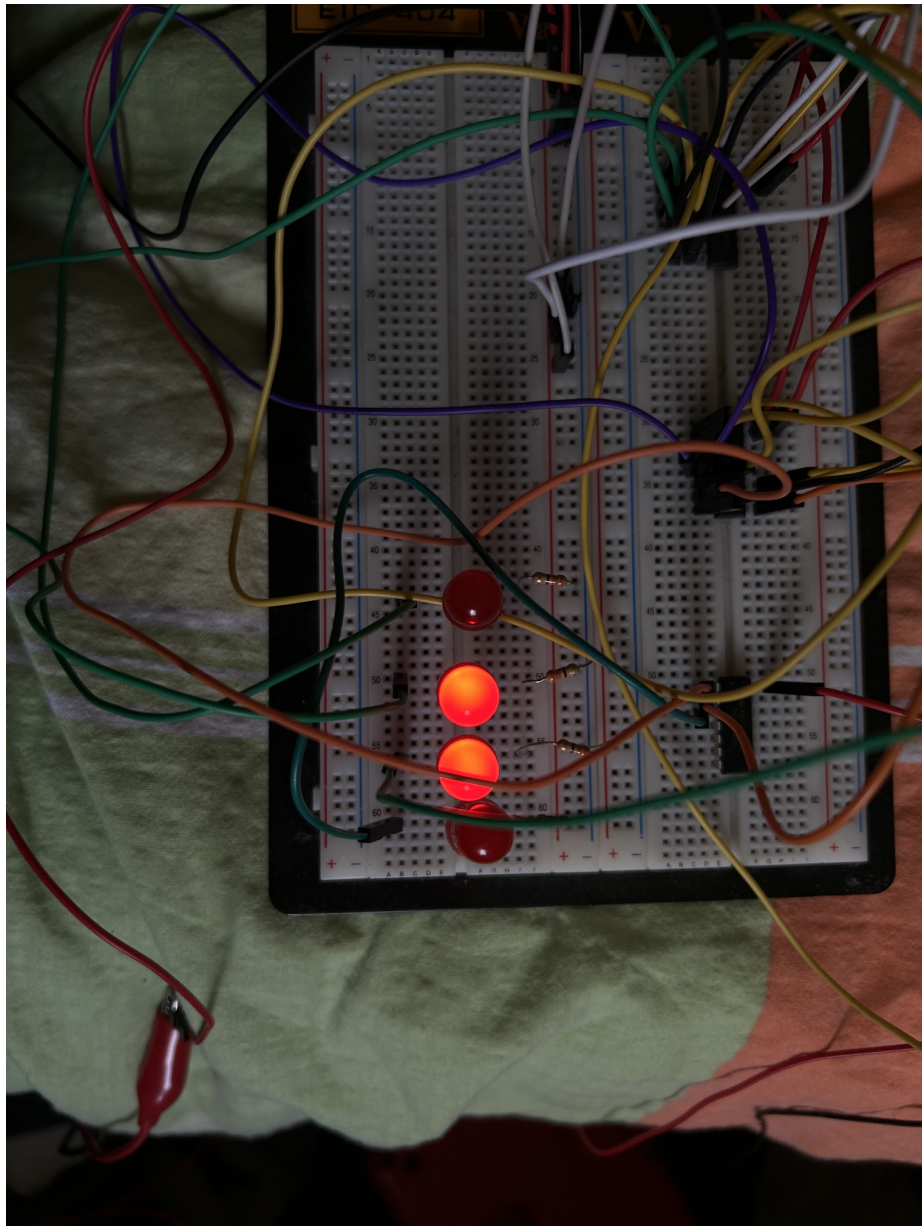


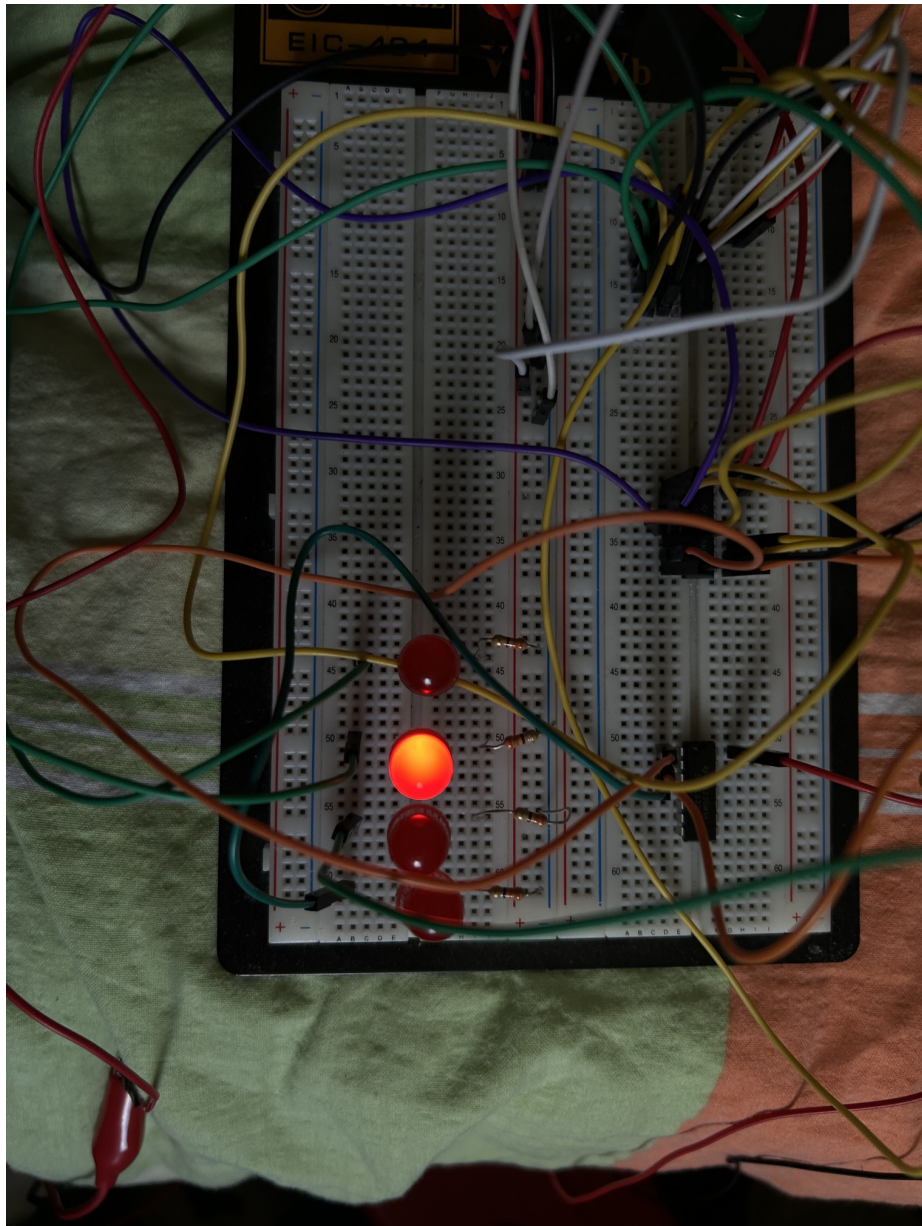


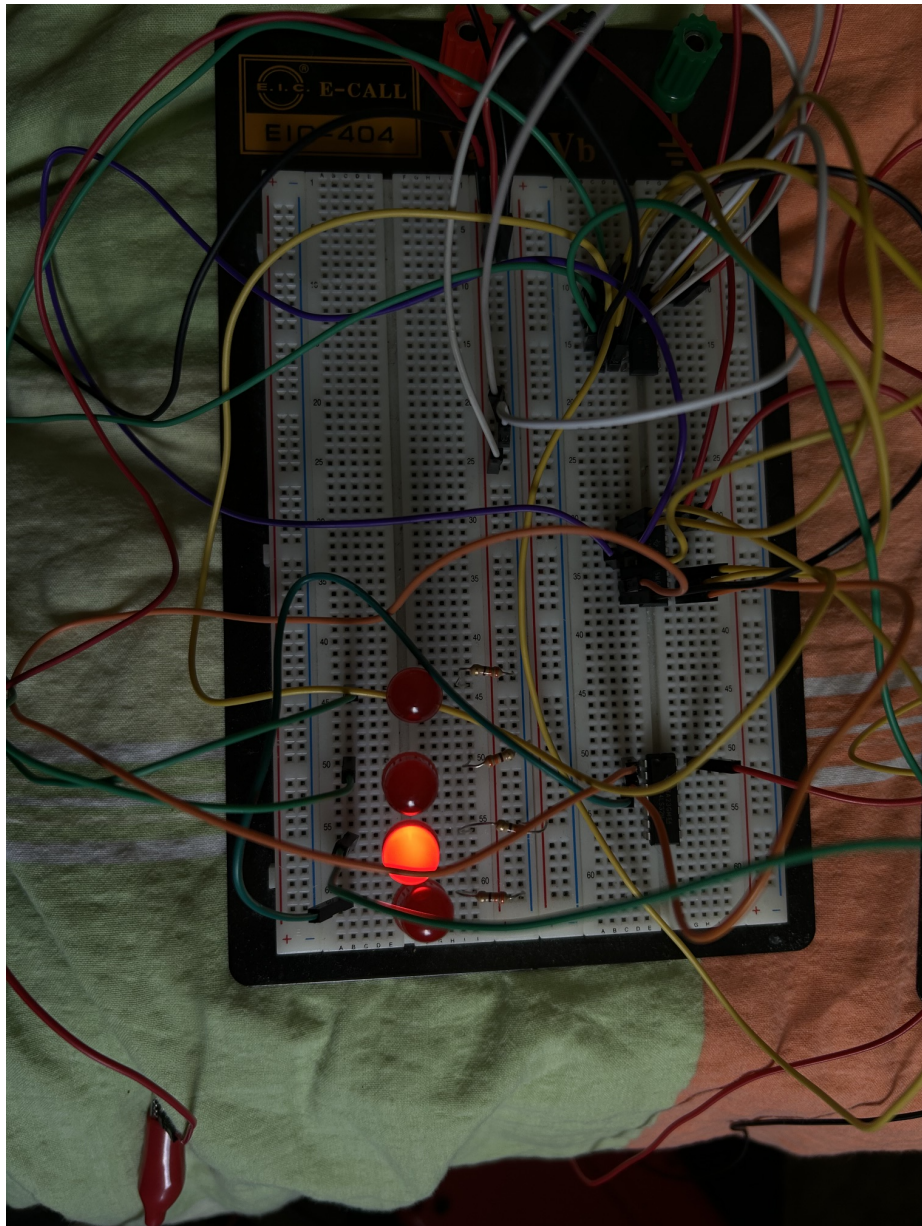


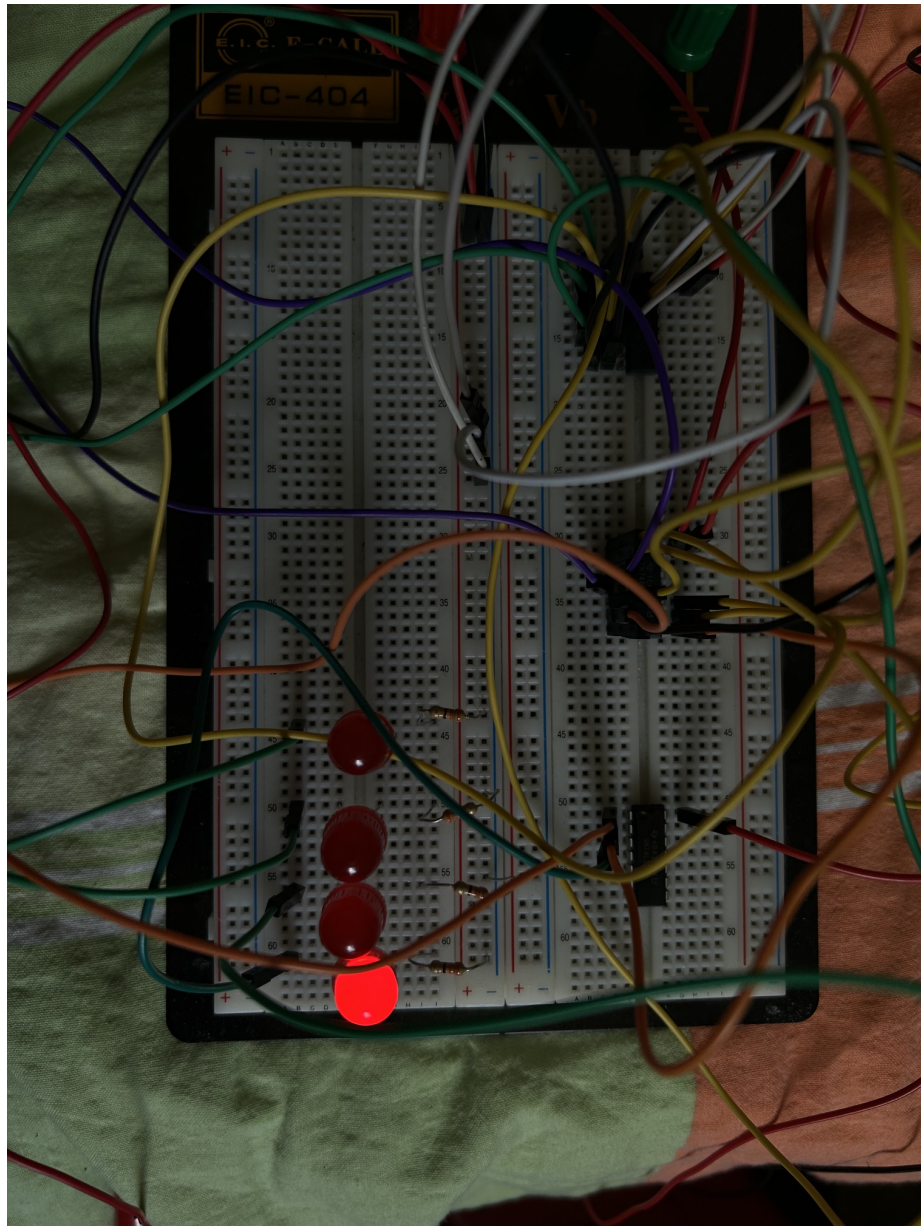












2 erstens, mit oszi, code, angepasst

```
;; (c) david vajda  
;; 07/21/2026  
;; m8 oszi count, wait loop  
  
.include "m8def.inc"  
  
ldi r16, HIGH (RAMEND)
```



```

out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16
ldi r16, 0xff
out DDRB, r16

```

```

ldi r16, 0x00
main001:
out PORTB, r16
rcall waitloop001
inc r16
rjmp main001

```

```

waitloop001:
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
ret

```

```

;; taktlaengen: main001: out PORTB, r16      ;; out:      1
;; rcall waitloop001                        ;; rcall:     3
;; inc r16                                  ;; inc:       1
;; rjmp main001                             ;; rjmp:      2

```

```

;; 1 + 3 + 1 + 2 = 7
;; man koennte rcall weg machen, es sind 4: und wg. 5: nop
;; aber nein:
;; ret                                     ;; ret:      4
;; zusammen: 11
;; und auf: 1MHz/20CLK = 50 kHz
;; wg. 11: 9 nop

```

```

;; jetzt time/div:
;; pro (1/50kHz) = (time/div) =
;; 2 * 5 = 10
;; 0.2 * 5 = 1
;; 0.02 * 50 = 1
;; also: 0.02, ...

```

3 timer intervall

```
;; (c) david vajda
;; 07/21/2026
;; m8 timer/counter interrupt

.include "m8def.inc"

.org 0x000
rjmp RESET
.org OVFIaddr
rjmp OVFIhndlr

RESET:
ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16
ldi r16, 0xff
out DDRB, r16

;; ldi r16, (CS)?? ics csi cis???
;; 2, 1, 0 und kombi

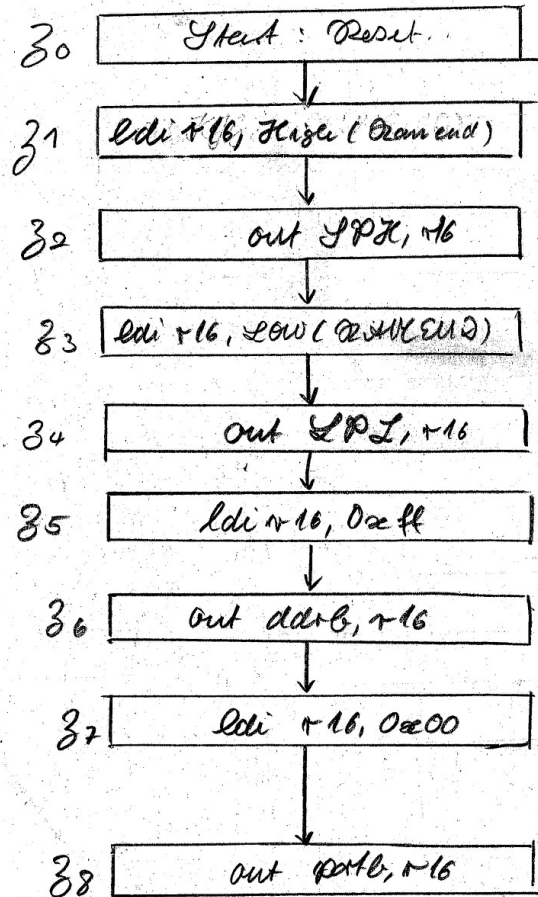
;; jetzt habe ich festgestellt, aber das wusste ich ja
;; CS: heisst, clock select, wie chip select, habe ich ja
;; runter gebetet. allerdings: verwechselt, externes interrupt...
ldi r16, (1 << CS12) | (1 << CS10)
out TCCR1A, r16
ldi r16, (1 << TOIE1)
out TIMSK, r16

sei

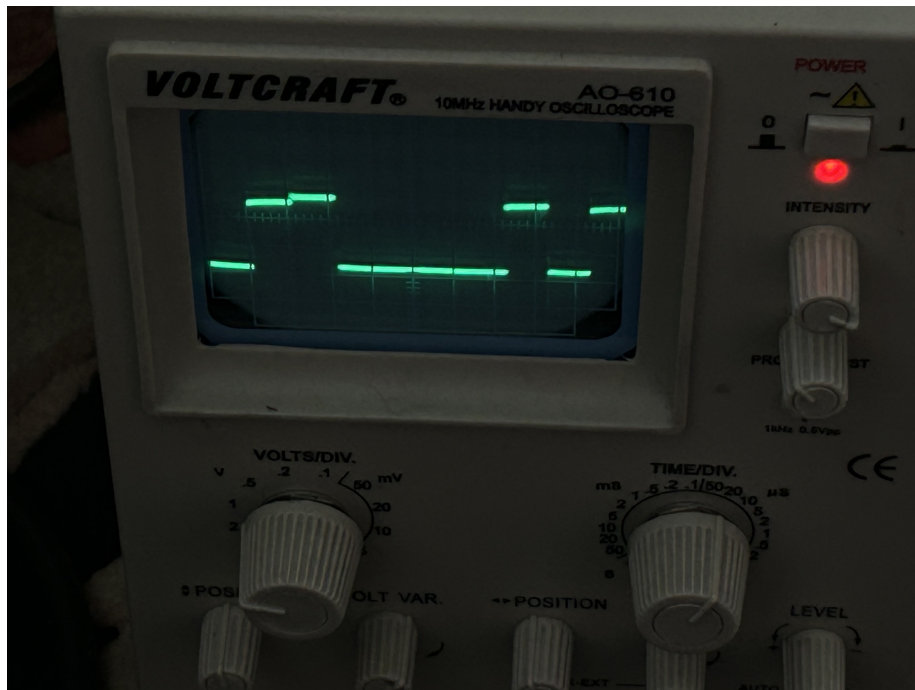
ldi r16, 0xff
out PORTB, r16
main001:  rjmp main001

OVFIhndlr:
dec r16
out PORTB, r16
reti
```

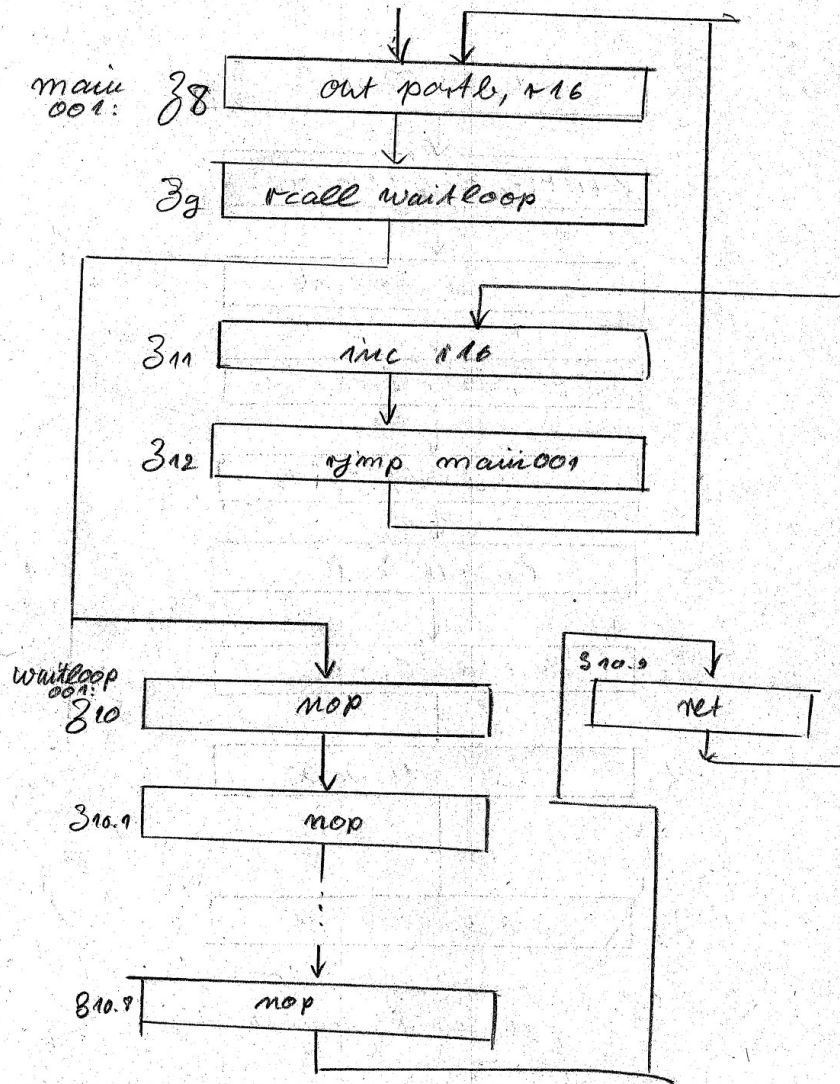
Saund Vojla (c)
07/21/2026
asm-diagramm Seite 1



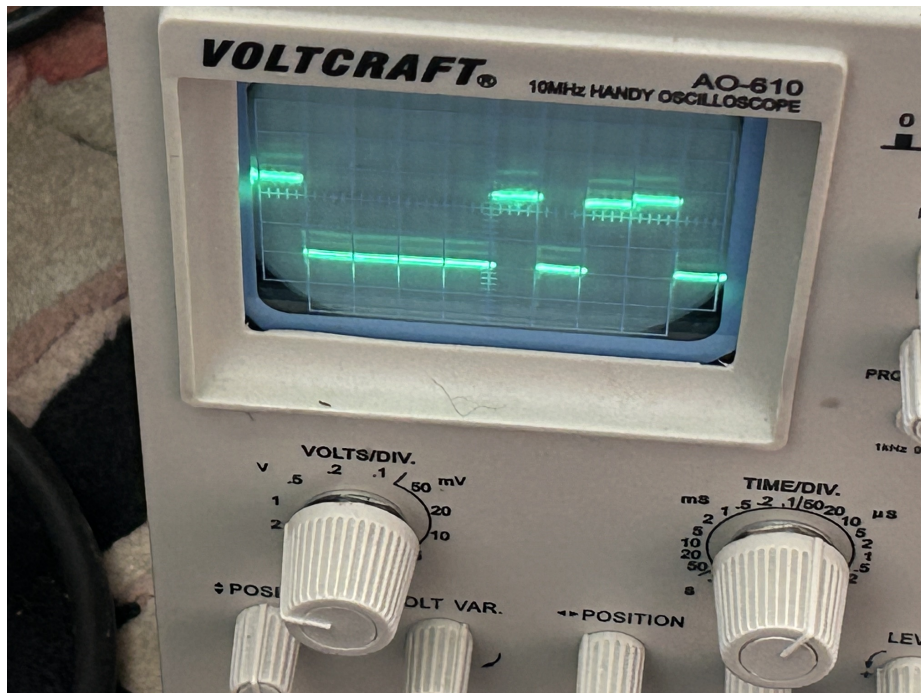
ob wdh. nächste Seite...

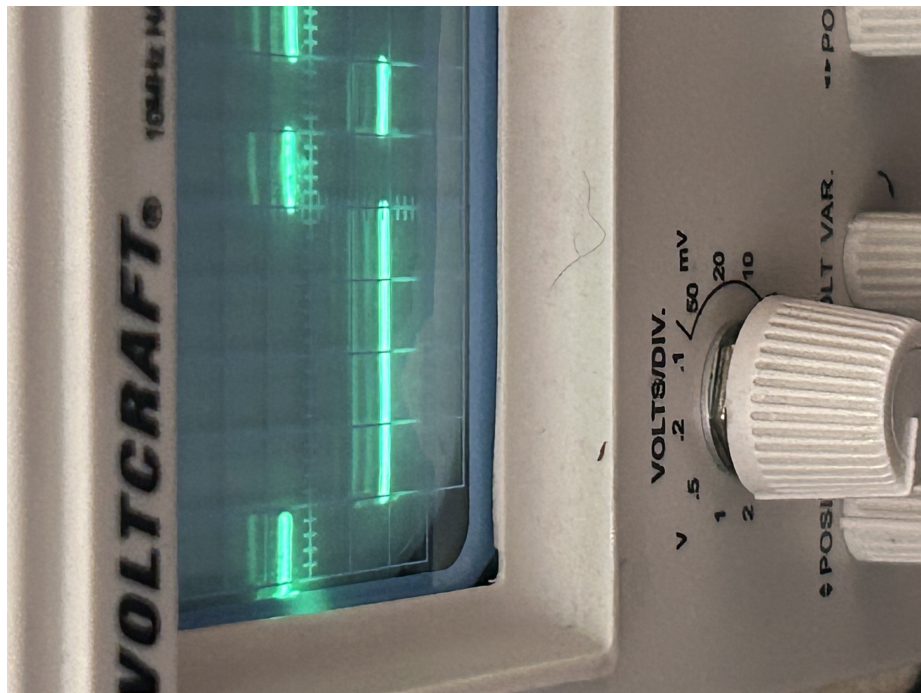


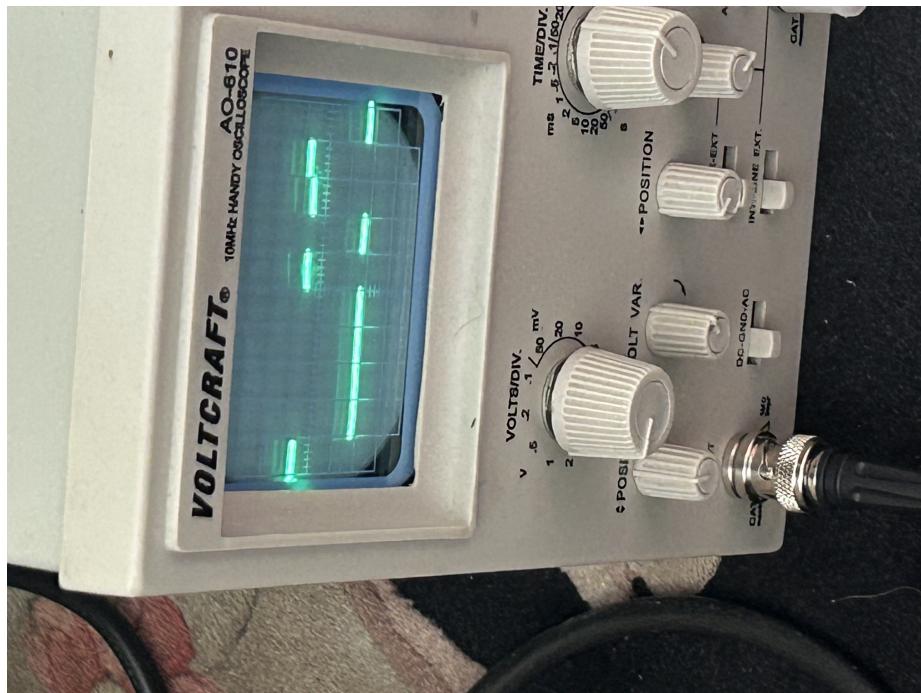
Diana Voyda, Seite 2
asm-Diagramm
07/21/2026

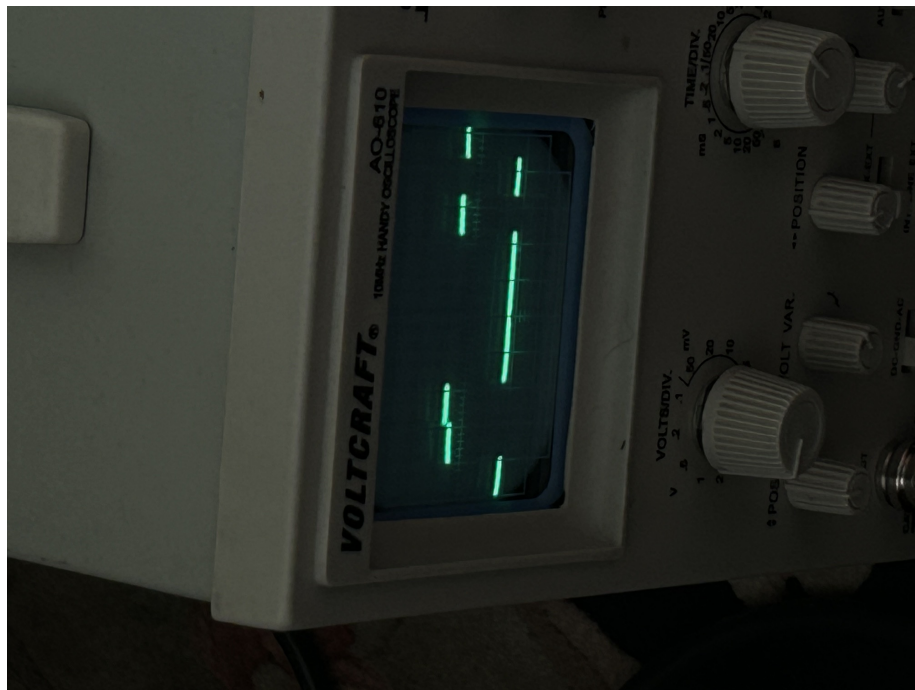












4 externe interrupts

das mit externen interrupts hat hier so funktioniert

```
;; (c) david vajda
;; 07/22/2022
;; externes interrupt

.include "m8def.inc"

.org 0x000
rjmp RESET
.org INT0addr
rjmp INT0hndlr
.org INT1addr
rjmp INT0hndlr

RESET:
ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, 0xff
out DDRB, r16
ldi r16, 0xff
out PORTB, r16
```

```
ldi r16, (1 << ISC11) | (1 << ISC10)
out MCUCR, r16
```

```
ldi r16, (1 << INT1) | (1 << INTO)
out GICR, r16
```

```
sei
main001: rjmp main001
```

```
INT0hndlr:
dec r16
out PORTB, r16
reti
```

```
INT1hndlr:
inc r16
out PORTB, r16
reti
```

photos folgen

5 das timerinterrupt

zeigt leider derzeit keine wirkung

```
;; (c) david vajda
;; 07/21/2026
;; m8 timer/counter interrupt
```

```
.include "m8def.inc"
```

```
.org 0x000
rjmp RESET
.org OVFIaddr
rjmp OVFIhndlr
```

```
RESET:
ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16
ldi r16, 0xff
out DDRB, r16
```

```
;; ldi r16, (CS)?? ics csi cis???
;; 2, 1, 0 und kombi
```

```

;; jetzt habe ich festgestellt, aber das wusste ich ja
;; CS: heisst, clock select, wie chip select, habe ich ja
;; runter gebetet. allerdings: verwechselt, externes interrupt...
ldi r16, (1 << CS12) | (1 << CS10)
out TCCR1A, r16
ldi r16, (1 << TOIE1)
out TIMSK, r16

sei

ldi r16, 0xff
;; out PORTB, r16
main001: rjmp main001

OVF1hndlr:
dec r16
out PORTB, r16
reti

```

6 rs232transmit

das tut so weit, allerdings zeichen nicht optimal, wissen:

```

USBS: USART STOP BIT SELECT
UCSZ2..0: USART CHARACTER SIZE (ich weiss nicht, 2 ist in C und 1..0 in B)... aber nicht j
UPM1:0 USART PARITY MODE, B oder C

```

hier der quelltext

```

;; (c) david vajda
;; 07/22/2026
;; rs232/transmit

#include "m8def.inc"

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, HIGH (25)
out UBRRH, r16
ldi r16, LOW (25)
out UBRRL, r16

ldi r16, (1 << URSEL)
out UCSRC, r16
ldi r16, (1 << TXEN)
out UCSRB, r16

```



```
rs232transmit001:
sbis UCSRA, UDRE
rjmp rs232transmit001
out UDR, r16
ret
```

```

Datei  Bearbeiten  Ansicht  Lesezeichen  Module  Einstellungen  Hilfe

Neues Unterfenster  Ansicht teilen  Kopieren  Einfügen  Suchen

$ E46 0000 0000
$ F.* 0000 ( 0000_.
*
-$ 00 00 00-
$ E46 0000 0000
$ F.* 0000 ( 0000_.
*
-$ 00 00 00-
$ E46 0000 0000
$ F.* 0000 ( 0000_.
*
-$ 00 00 00-
$ E46 0000 0000
$ F.* 0000 ( 0000_.
*
-$ 00 00 00-
^C
[1]+  Done                  stty -F /dev/ttyS3 2400
root@www001:/home/david#
```

```
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
📄 Neues Unterfenster 🗑️ Ansicht teilen ▾ 🔍 Kopieren 📋 Einfügen 🔍 Suchen
```

```
$ E46 F.* ( ) .
$ F.* ( ) .
-$ -
E46 F.* ( ) .
$ F.* ( ) .
-$ -
E46 F.* ( d) ( ) .
-$ -
E46 F.* ( ) .
-$ -
E46 F.* ( ) .
-$ -
```

```

;; 07/22/2026
;; rs232/transmit

.include "m8def.inc"

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16

ldi r16, HIGH (25)
out UBRRH, r16
ldi r16, LOW (25)
out UBRRL, r16
ldi r16, 0xff
out DDRB, r16

ldi r16, (1 << URSEL)
out UCSRC, r16
ldi r16, (1 << RXEN)
out UCSRB, r16

main001:
rcall rs232recievech001
out PORTB, r16
rjmp main001

rs232recievech001:
sbis UCSRA, UDRE
rjmp rs232recievech001
in r16, UDR
ret

einigermassen:
#!/bin/bash

# (c) david vajda
# 07/23/2026
# rs232 recieve (set) from (transmit) terminal

i=0
M=(A B C D E F G H I J M N O P Q R S T U V W X Y Z)
i=0
while [ $i -lt "${#M[@]}" ]
do
    stty -F /dev/ttyS3 2400 & echo "${M[$i]}" >> /dev/ttyS3
    sleep 2
    i=$((i+1))
done

```

8 lcd hd44780

```
;; (c) david vajda
;; 07/22/2026
;; lcd44780

.include "m8def.inc"

ldi r16, HIGH (RAMEND)
out SPH, r16
ldi r16, LOW (RAMEND)
out SPL, r16
ldi r16, 0xff
out DDRB, r16

rcall hd44780init001

main001:
ldi ZH, HIGH (2*msg)
ldi ZL, LOW (2*msg)
hd44780mainloop001:
lpm r16, Z+
cpi r16, 0x00
breq hd44780mainloop001
rcall hd44780putc001
rcall waitingloop1s001
rjmp main001

msg: .db "david@https://www.vajda-breadboard.de", 10, 13, 0

hd44780init001:
ldi r16, 50
hd44780init001loop001:
rcall hd44780waitlooplong001
dec r16
brne hd44780init001loop001
ldi r16, 0x03
rcall hd44780waitlooplong001
ldi r16, 0x03
rcall hd44780waitloopshort001
ldi r16, 0x03

ldi r16, 0x02
out PORTB, r16
rcall hd44780enable001
rcall hd44780waitlooplong001

ldi r16, 0x20
rcall hd44780cmd001
rcall hd44780enable001
```

```
rcall hd44780waitlooplong001
ret
```

```
hd44780waitlooplong001:
push r16
push r17
ldi r16, 0x21
hd44780waitlooplong001l001:
ldi r17, 0xC9
hd44780waitlooplong001l002:
dec r17
breq hd44780waitlooplong001l002
dec r16
breq hd44780waitlooplong001l001
pop r17
pop r16
ret
```

```
hd44780waitloopshort001:
push r16
ldi r16, 42
hd44780waitloopshort001l001:
dec r16
breq hd44780waitloopshort001l001
pop r16
ret
```

```
hd44780enable001:
sbi PORTB, 5
nop
nop
nop
cbi PORTB, 5
ret
```

```
hd44780putc001:
mov r17, r16
swap r16
andi r16, 0x0f
out PORTB, r16
rcall hd44780enable001
andi r17, 0x0f
out PORTB, r17
rcall hd44780enable001
ret
```

```
hd44780cmd001:
mov r17, r16
swap r16
andi r16, 0x0f
```



```

ori r16, 0b00010000
out PORTB, r16
rcall hd44780enable001
andi r17, 0x0f
ori r17, 0b00010000
out PORTB, r17
rcall hd44780enable001

```

```
ret
```

```

waitingloop1s001:
push r16
push r17
push r18
ldi r18, 0x04
waitingloop1s1001:
ldi r17, 0xff
waitingloop1s1002:
ldi r16, 0xff
waitingloop1s1003:
dec r16
breq waitingloop1s1003
dec r17
breq waitingloop1s1002
dec r18
breq waitingloop1s1003
pop r18
pop r17
pop r16
ret

```

tut gerade noch nicht...