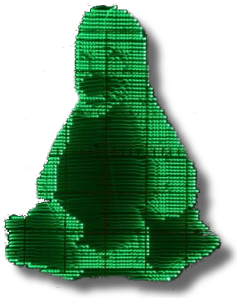
„Wenn Geeks Langeweile haben“ - reloaded

Uwe Berger

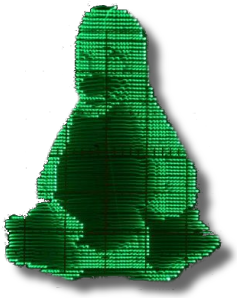
bergeruw@gmx.net



Uwe Berger

- Beruf: Softwareentwickler
- Freizeit: u.a. mit Hard- und Software rumspielen
- Linux seit ca. 1995
- BraLUG e.V.
- bergeruw@gmx.net

Rückblick CLT2013...



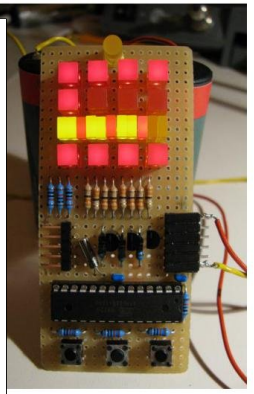
Wenn Geeks Langeweile haben...

RGB-LED

- Motivation:
  - „Wie erzeugt man Farbtöne
    einer RGB-LED?“
- Hardware:
  - Mikrocontroller: Atmel ATti

Wenn Geeks Langeweile haben...

Berliner Uhr



22

Wenn Geeks Langeweile haben...

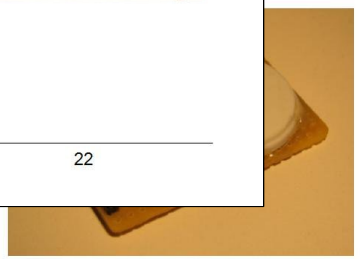
1aus6-LED-Würfel

- Motivation:
  - „Wie programmiert man
  - „Wie funktioniert Multiple
- Hardware:
  - Mikrocontroller: Atmel A
  - 7 LEDs, 1 Taster
- Software:
  - C (avrgcc)
  - Zufallsgenerator, Multipl
- Dokumentation:
  http://bralug.de/wiki/ATtiny-Sonntagsspielereien#1aus6-W.C3.BCrfe

Uwe Berger; CLT2013 12

Wenn Geeks Langeweile haben...

Uwe Berger
bergeruw@gmx.net

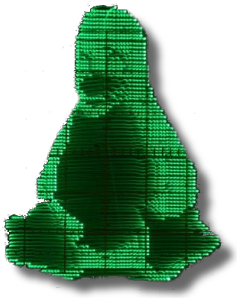


g, Sleep-Mode

- Dokumentation:
  http://bralug.de/wiki/ATtiny-Sonntagsspielereien#Melodiegenerator

Uwe Berger; CLT2013 17

Rückblick CLT2013...



Wenn Geeks Langeweile haben...

RGB-LED

- Motivation:
  - „Wie erzeugt man einer RGB-LED?“
- Hardware:
  - Mikrocontroller: At

Wenn Geeks Langeweile haben...

Berliner Uhr

22

Wenn Geeks Langeweile haben

1aus6-LED-W

- Motivation:
  - „Wie programmi
  - „Wie funktionier
- Hardware:
  - Mikrocontroller:
  - 7 LEDs, 1 Taste
- Software:
  - C (avrgcc)
  - Zufallsgenerator
- Dokumentation:
  http://bralug.de/wiki/ATtiny-Sonntagsspielereien#1aus6-W.C3.BCrfe

Uwe Berger; CLT2013 12

-Mode

- Dokumentation:
  http://bralug.de/wiki/ATtiny-Sonntagsspielereien#Melodiegenerator

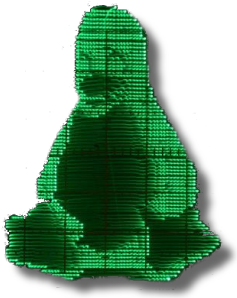
Uwe Berger; CLT2013 17

<u>Kritikpunkte im Nachgang:</u>

„...das war zu viel auf einmal!“

„...technische Hintergründe/Grundlagen wurden zu kurz angerissen!“

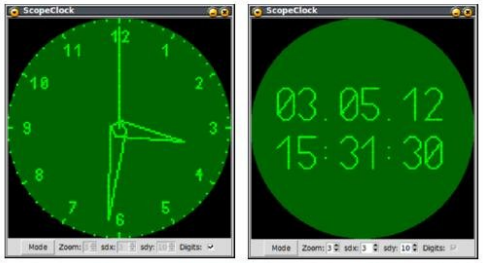
Rückblick CLT2013...



Wenn Geeks Langeweile haben...

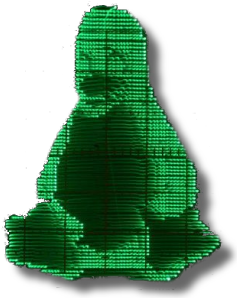
Projektvorhaben: Scopeclock

- Motivation:
  - „Einfach nur faszinierend...!“
- Hardware:
  - Kathodenstrahlröhre
  - Mikrocontroller: ???
  - ein 600V-Netzteil (!)
- Software:
  - C und/oder Assembler
  - ein Simulator (Tcl/Tk)
- Dokumentation:
  http://bralug.de/wiki/Scopeclock



Uwe Berger; CLT2013 25

Rückblick CLT2013...



Wenn Geeks Langeweile haben...

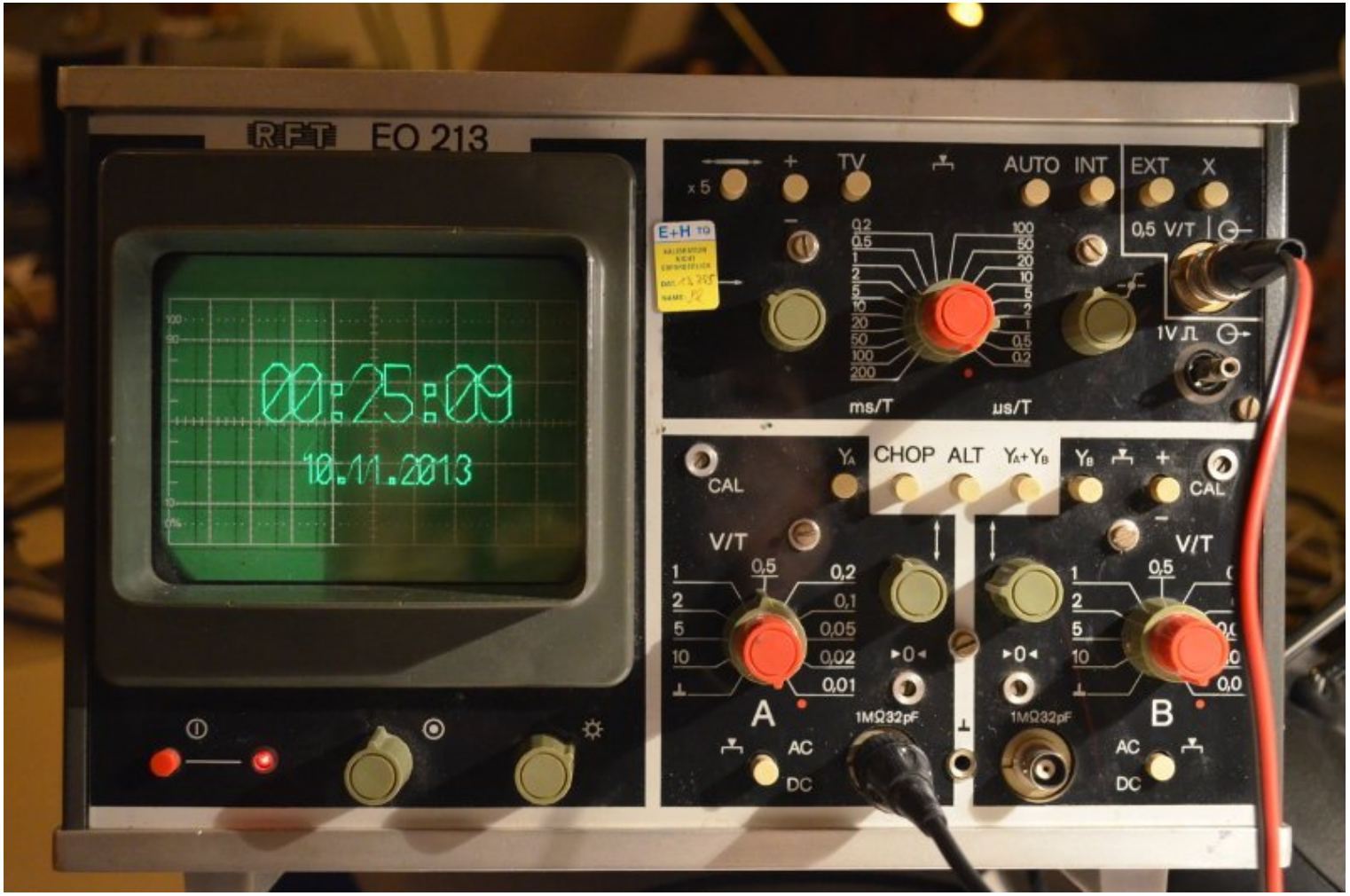
Projektvorhaben: Scopeclo

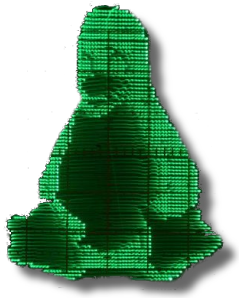
- Motivation:
  - „Einfach nur faszinierend...!“
- Hardware:
  - Kathodenstrahlröhre
  - Mikrocontroller: ???
  - ein 600V-Netzteil (!)
- Software:
  - C und/oder Assembler
  - ein Simulator (Tcl/Tk)
- Dokumentation:
  http://bralug.de/wiki/Scopeclock

„Nehme doch erst mal ein analoges Oszilloskop...“

Uwe Berger; CLT2013 25

Rückblick CLT2013...



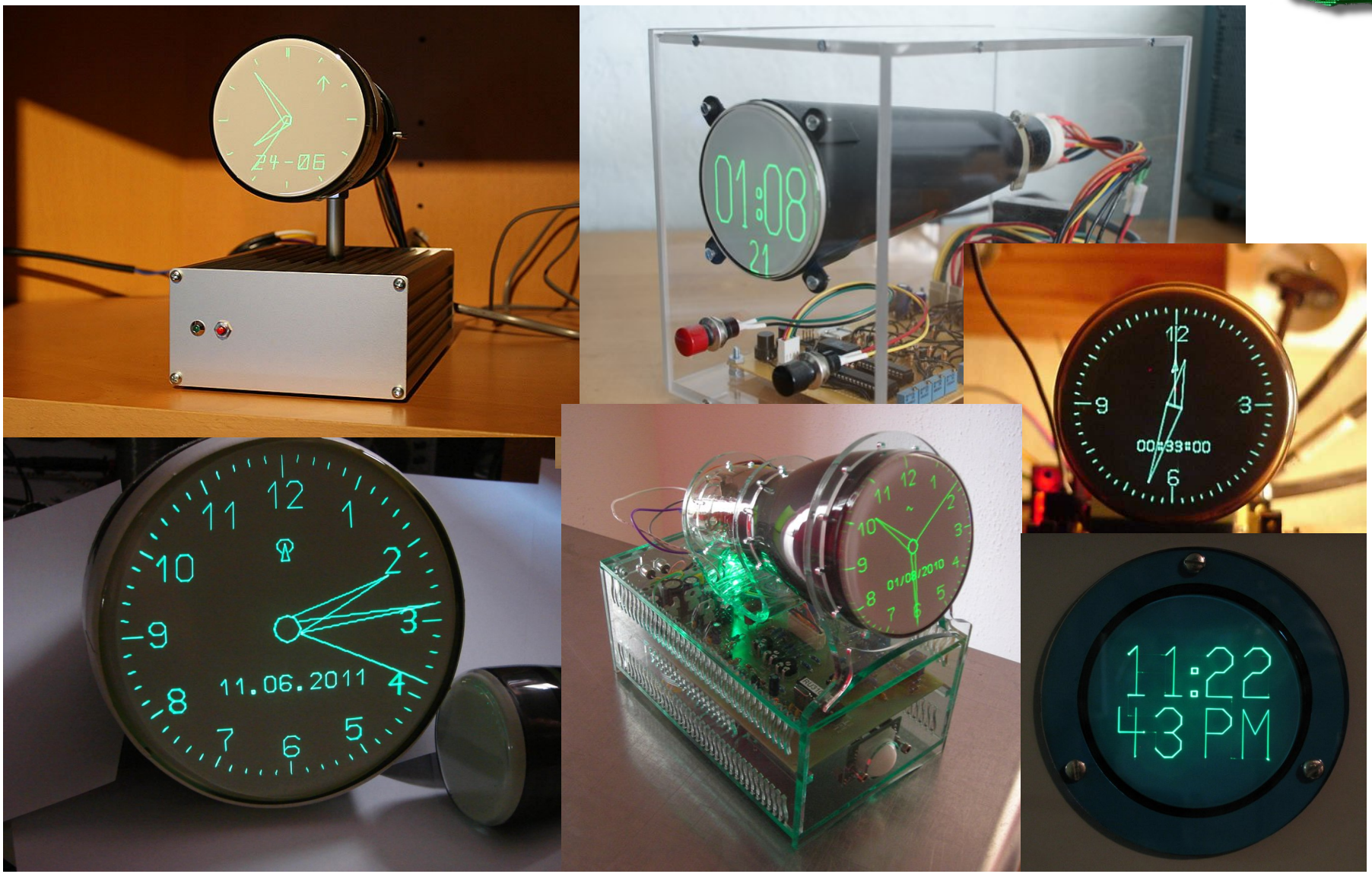


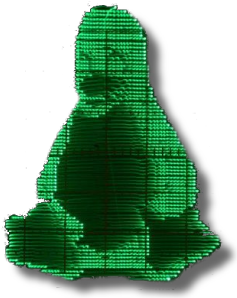
Inhalt

Schritte zum eigenen Mikrocontrollerprojekt am Beispiel „Scopeclock“:

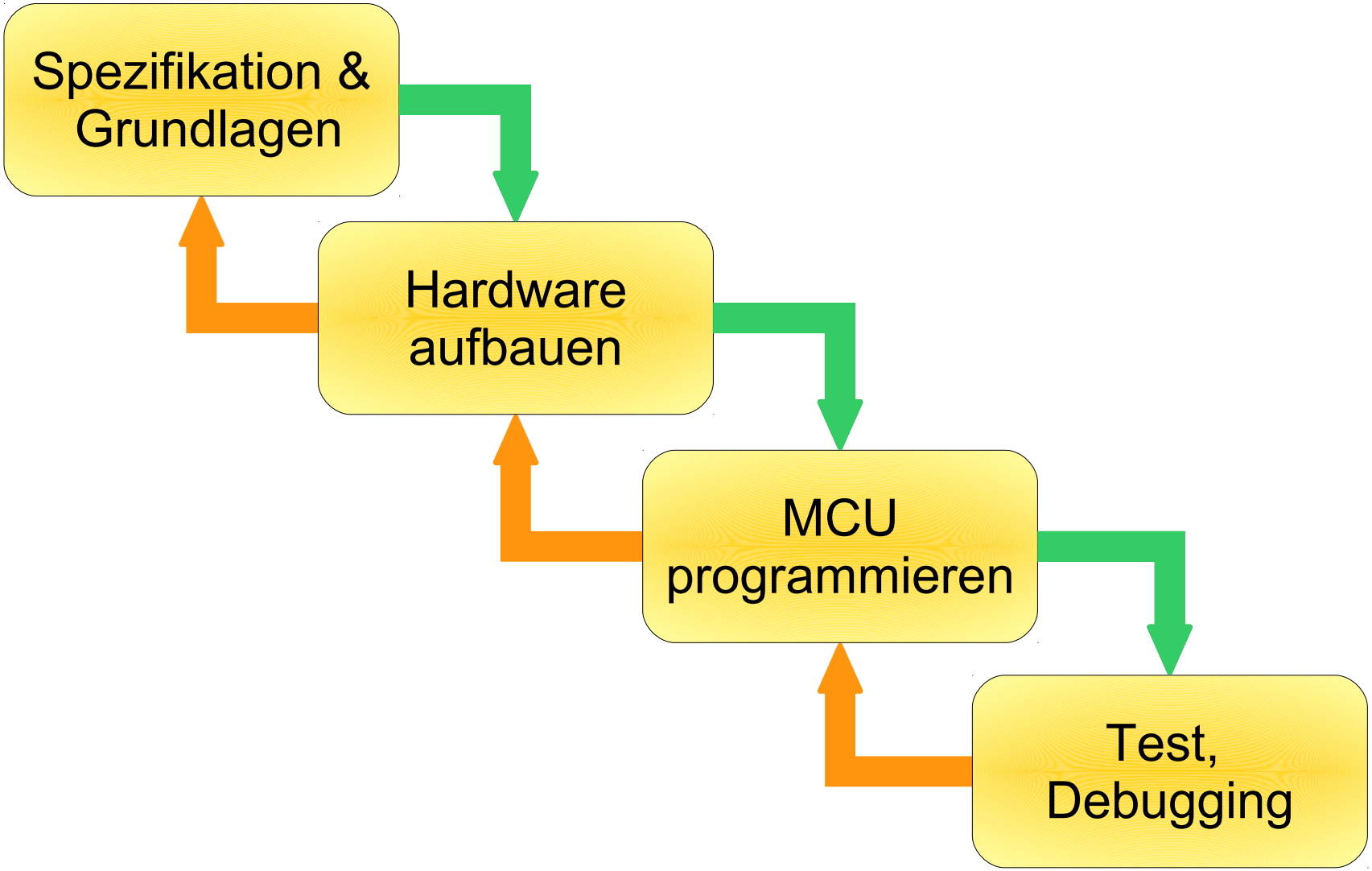
- Spezifikation und Grundlagen
- Hardware
- Software
- Test und Fehlersuche

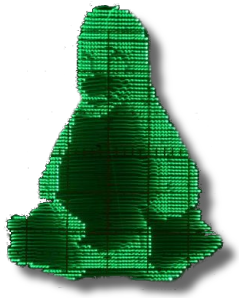
„Ich habe eine Idee...“





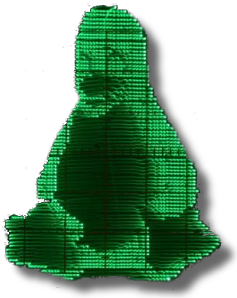
Schritte zum eigenen Projekt





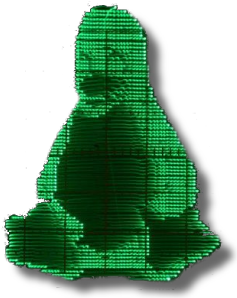
Spezifikation & Grundlagen

- Was soll das Ding tun?
- Wie soll das Ding es tun?
- Wie sehen die technischen Grundlagen/Verfahren dazu aus?
- Welche Grenzen gibt es und was sind die Konsequenzen daraus?



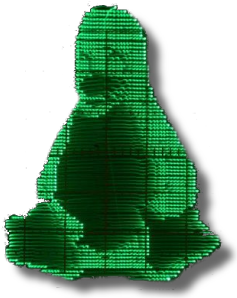
Was soll die „Scopeclock“ können?

- Die Uhrzeit auf einer Kathodenstrahlröhre anzeigen
- Uhrzeit soll möglichst genau, einstellbar sein und gepuffert werden
- Ein paar Gimmicks:
  - Bilder anzeigen,
  - Bildschirmschoner



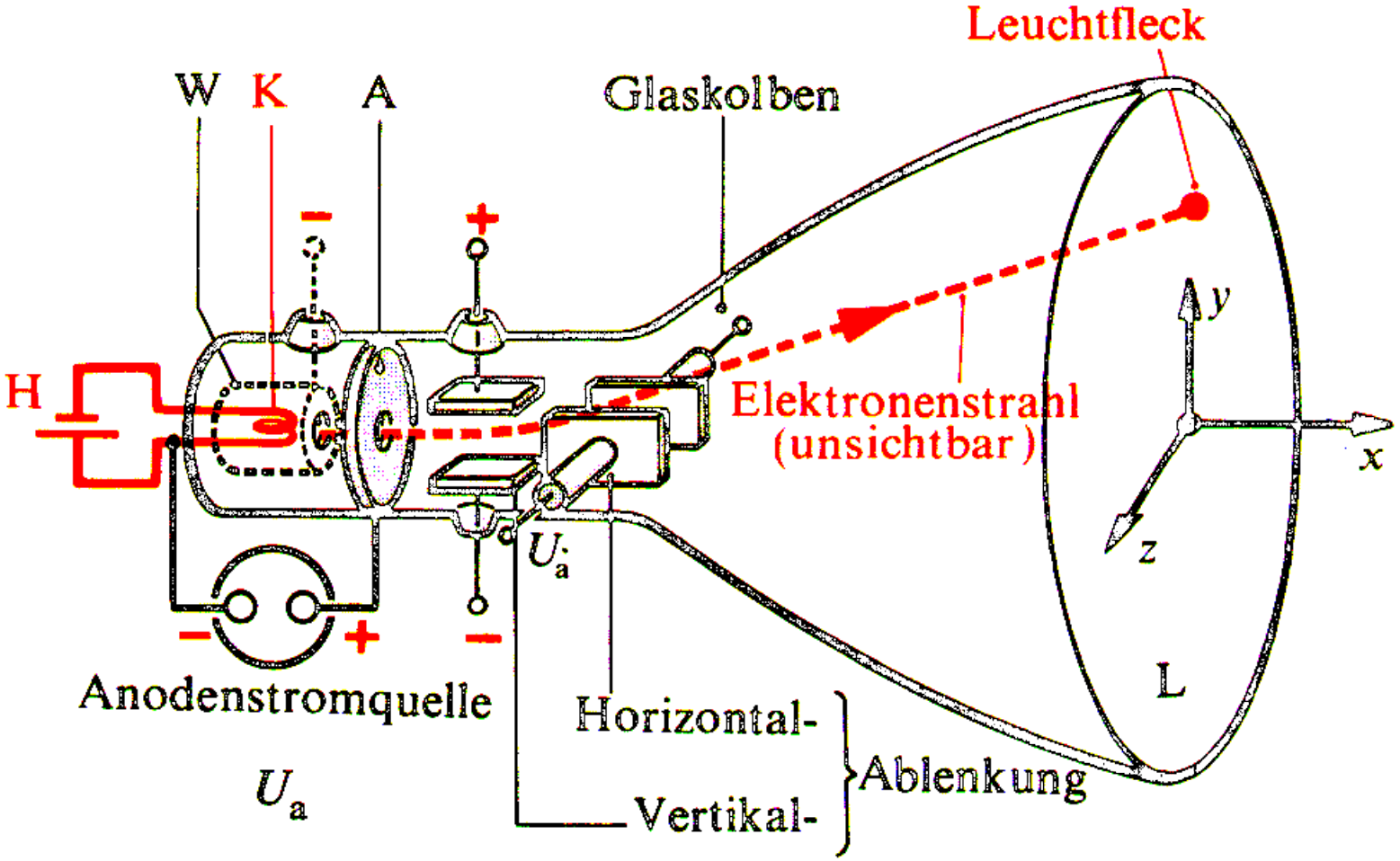
Grundlagen zur „Scopeclock“

- Wie funktioniert eine Kathodenstrahlröhre?
- Wie steuert man eine Kathodenstrahlröhre „punktgenau“ an?
- Wie zeichnet man Linien, Kreise und Zeichen?
- Wie generiert man einen möglichst genauen Sekunden-takt?
- Wie zählt man die Uhrzeit im ausgeschalteten Zustand weiter?

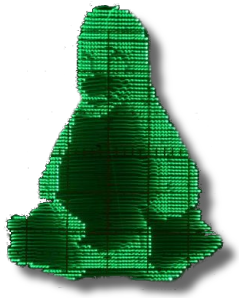


Kathodenstrahlröhre

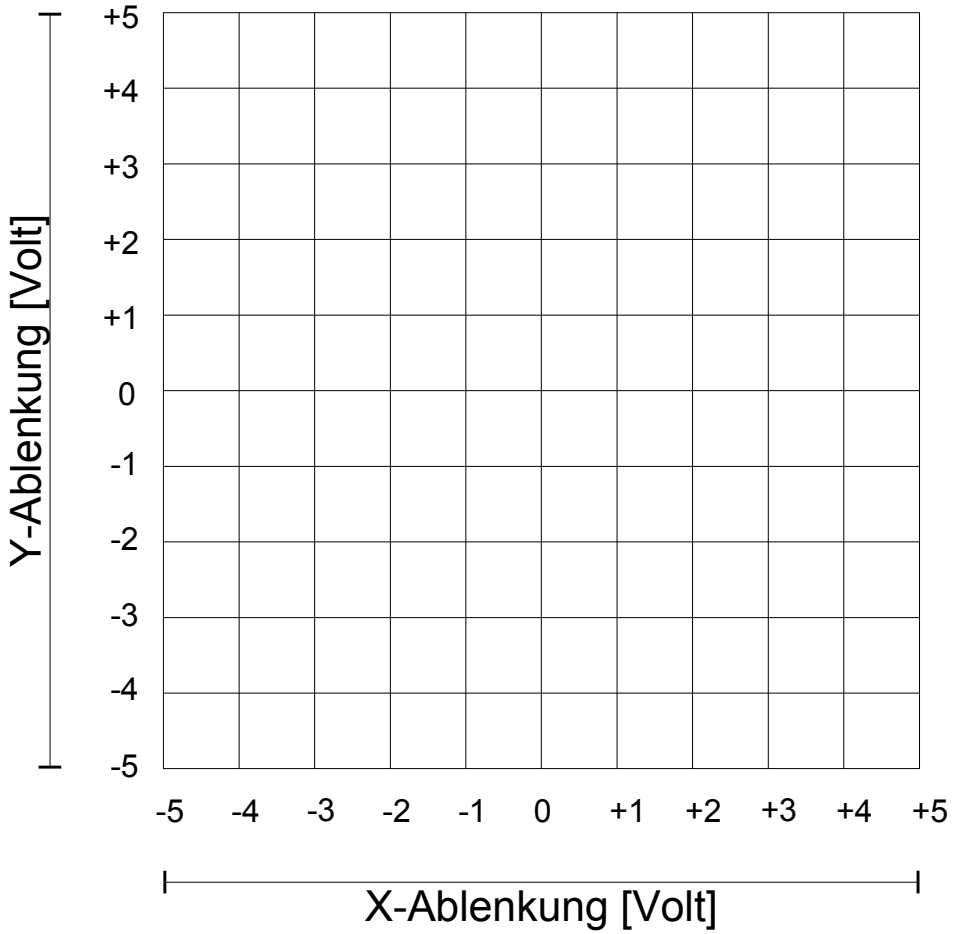
Kathodenstrahlröhre

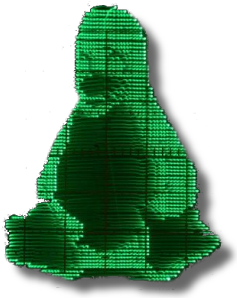


Quelle: http://www.physikmathe.de/



X-/Y-Ablenkung

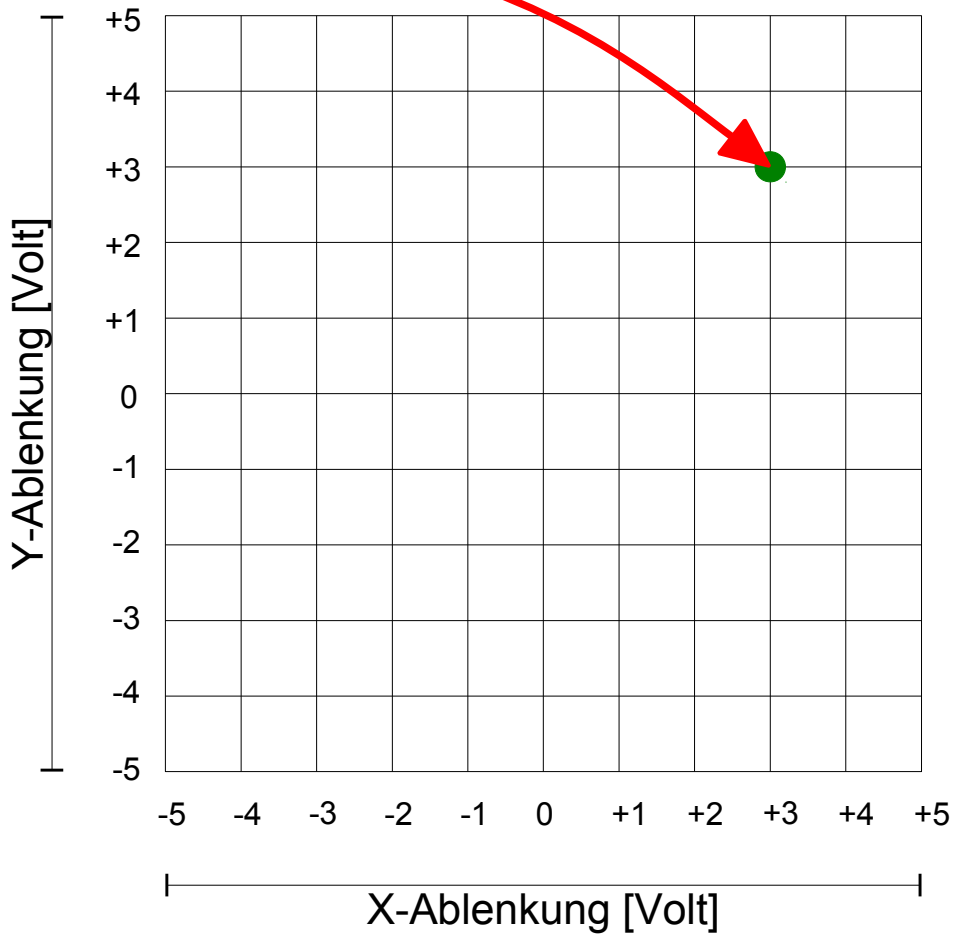




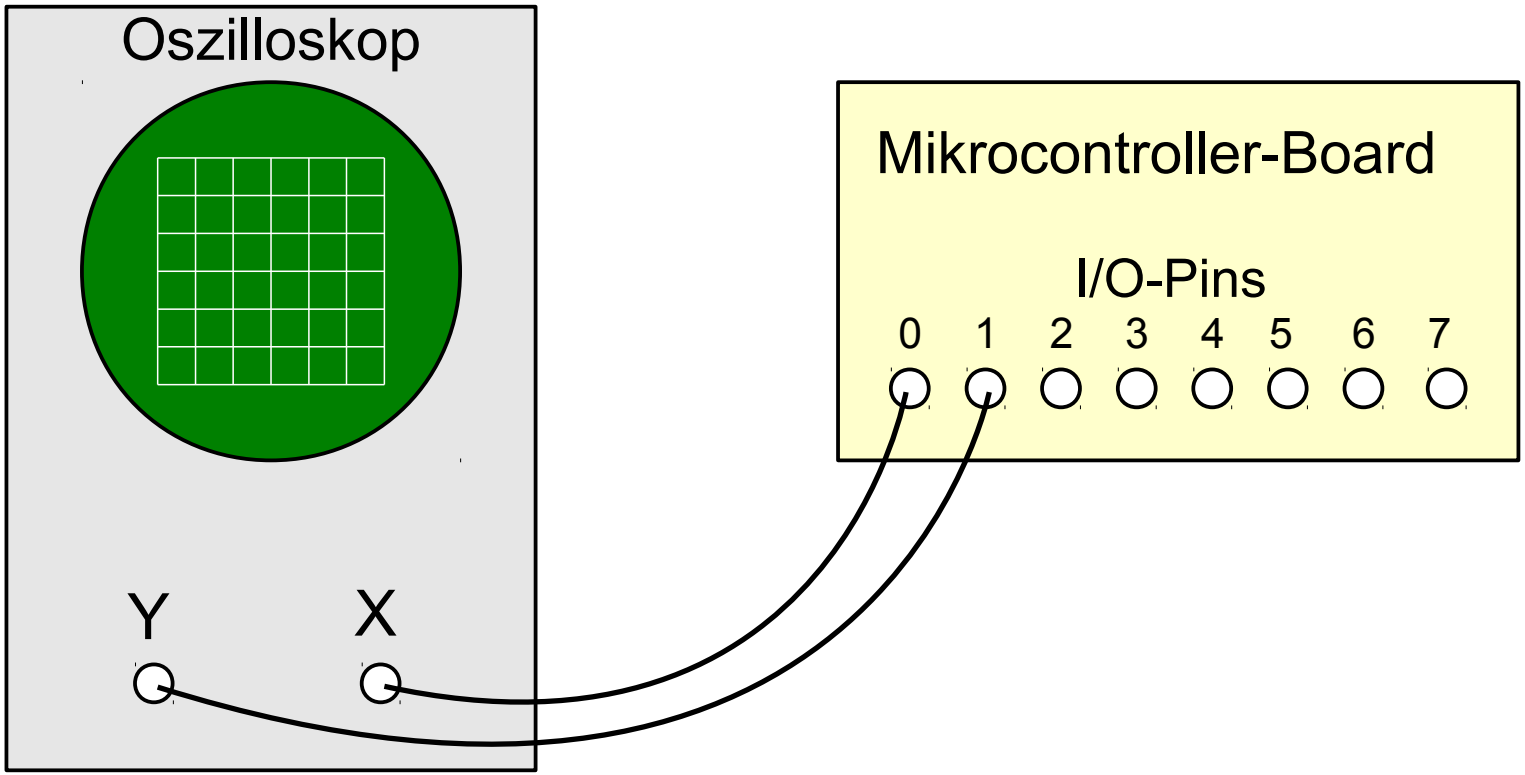
X-/Y-Ablenkung (...ein Punkt)

Ablenkspannung:

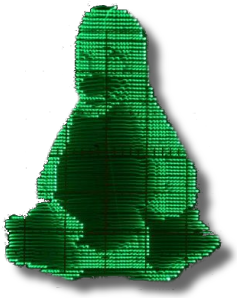
- X-Richtung → 3V
- Y-Richtung → 3V



Erste Punkte zeichnen (Versuchsaufbau)



Erste Punkte zeichnen



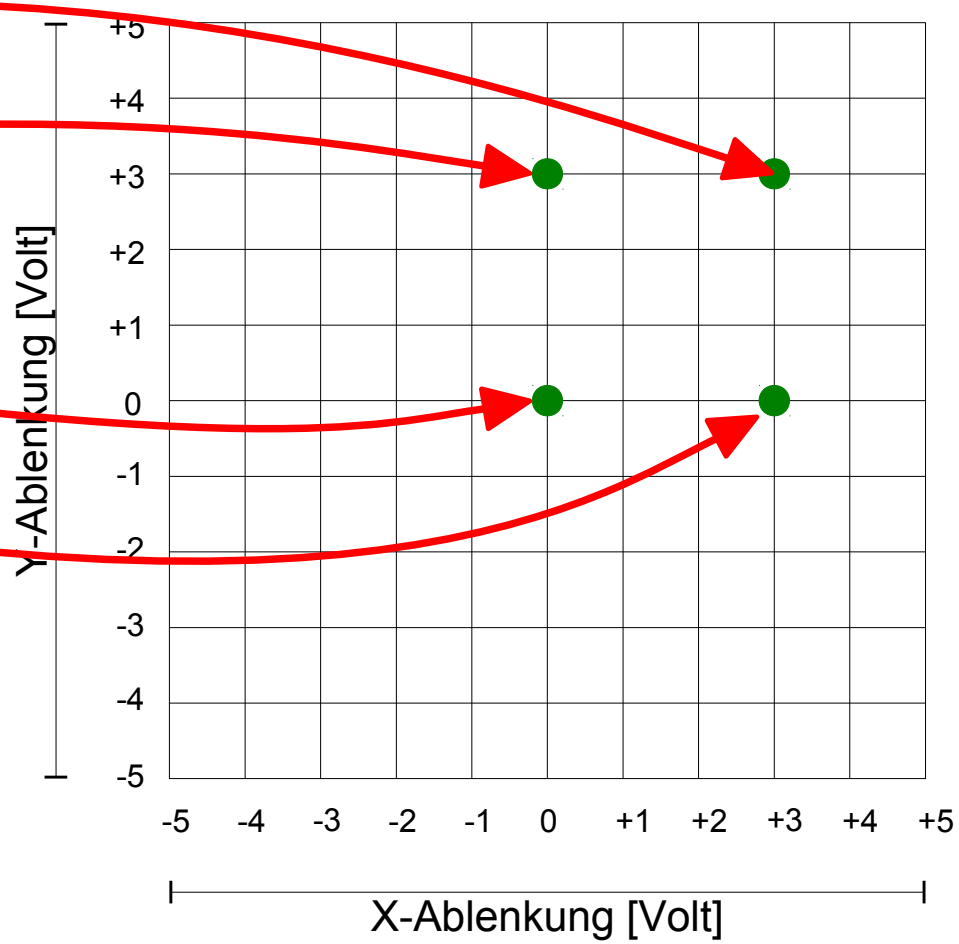
Ablenkspannung (X/Y):

→ 3V/3V

→ 0V/3V

→ 0V/0V

→ 3V/0V

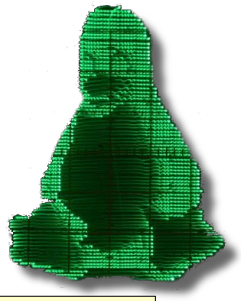


I/O-Pin:

→ High (1) → ca. 3V

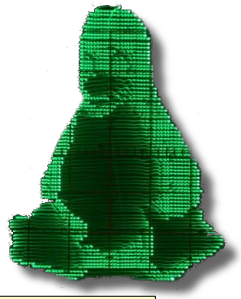
→ Low (0) → ca. 0V

Erste Punkte zeichnen (MCU-Programm)



```
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"

#define XY_PORT_BASE      GPIO_PORTB_BASE
#define XY_SYSCTL_PERIPH  SYSCTL_PERIPH_GPIOB
#define PIN_XOUT          GPIO_PIN_0
#define PIN_YOUT          GPIO_PIN_1
#define DELAY             100

int main()
{
    SysCtlPeripheralEnable(XY_SYSCTL_PERIPH);
    GPIOPinTypeGPIOOutput(XY_PORT_BASE, PIN_XOUT|PIN_YOUT);
    while (1) {
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, 0);                  // 0,0
        SysCtlDelay(DELAY);
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, PIN_XOUT);           // 1,0
        SysCtlDelay(DELAY);
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, PIN_XOUT|PIN_YOUT); // 1,1
        SysCtlDelay(DELAY);
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, PIN_YOUT);           // 0,1
        SysCtlDelay(DELAY);
    }
}
```

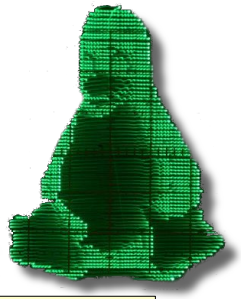
Erste Punkte zeichnen (MCU-Programm)



```
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"

#define XY_PORT_BASE      GPIO_PORTB_BASE
#define XY_SYSCTL_PERIPH  SYSCTL_PERIPH_GPIOB
#define PIN_XOUT          GPIO_PIN_0
#define PIN_YOUT          GPIO_PIN_1
#define DELAY             100

int main()
{
    SysCtlPeripheralEnable(XY_SYSCTL_PERIPH);
    GPIOPinTypeGPIOOutput(XY_PORT_BASE, PIN_XOUT|PIN_YOUT);
    while (1) {
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, 0);                  // 0,0
        SysCtlDelay(DELAY);
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, PIN_XOUT);           // 1,0
        SysCtlDelay(DELAY);
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, PIN_XOUT|PIN_YOUT); // 1,1
        SysCtlDelay(DELAY);
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, PIN_YOUT);           // 0,1
        SysCtlDelay(DELAY);
    }
}
```

Ausgabe-Pins für X- und Y-Richtung

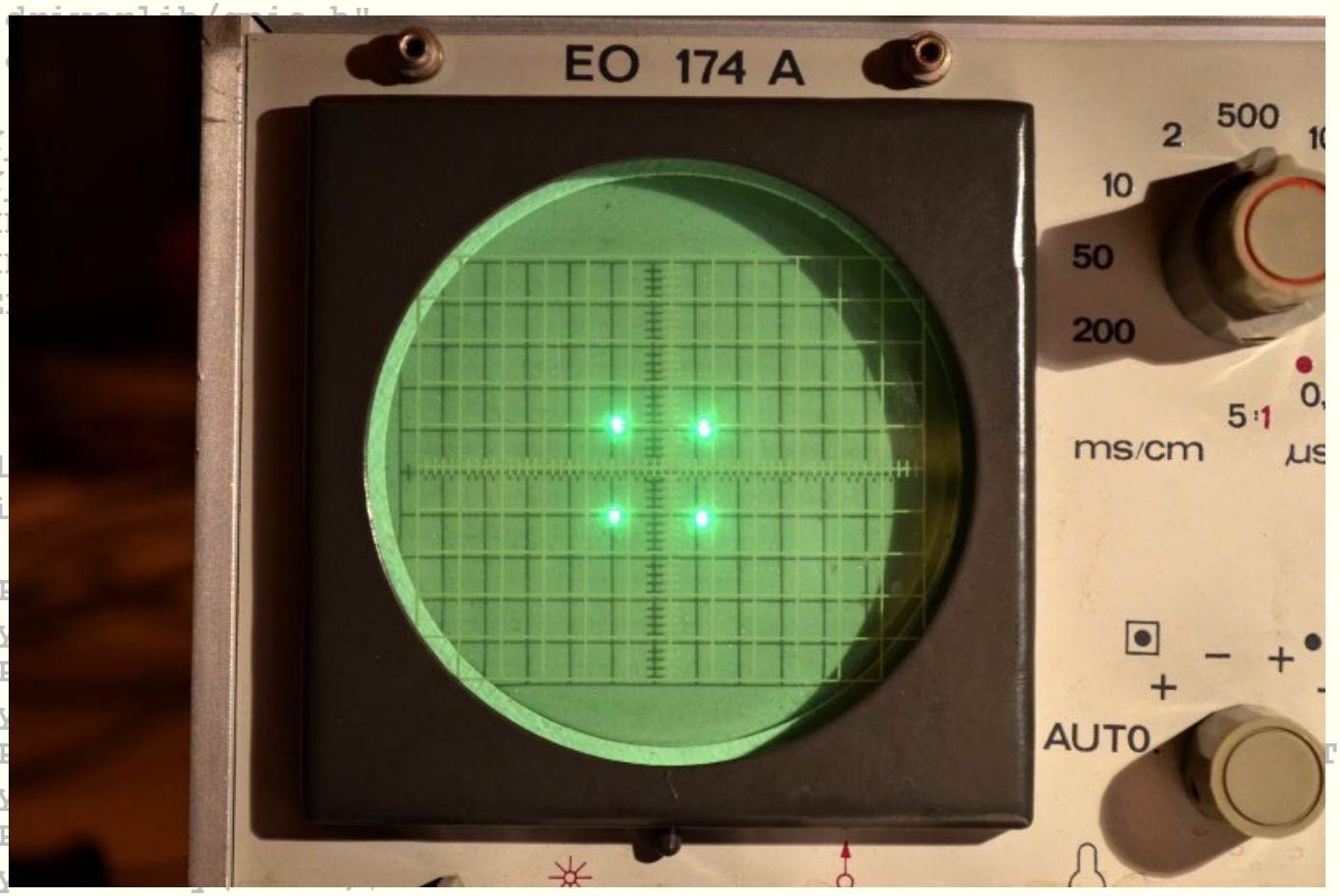
Erste Punkte zeichnen (MCU-Programm)



```
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"

#define XY_PORT_BASE      GPIO_PORTB_BASE
#define XY_SYSCTL_PERIPH  SYSCTL_PERIPH_GPIOB
#define PIN_XOUT          GPIO_PIN_0
#define PIN_YOUT          GPIO_PIN_1
#define DELAY             100

int main()
{
    SysCtlPeripheralEnable(XY_SYSCTL_PERIPH);
    GPIOPinTypeGPIOOutput(XY_PORT_BASE, PIN_XOUT|PIN_YOUT);
    while (1) {
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, 0);                 // 0,0
        SysCtlDelay(DELAY);
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, PIN_XOUT);          // 1,0
        SysCtlDelay(DELAY);
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, PIN_XOUT|PIN_YOUT); // 1,1
        SysCtlDelay(DELAY);
        GPIOPinWrite(XY_PORT_BASE, PIN_XOUT|PIN_YOUT, PIN_YOUT);          // 0,1
        SysCtlDelay(DELAY);
    }
}
```

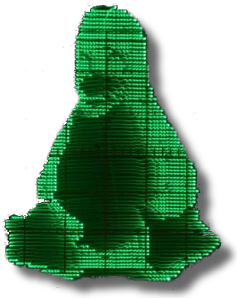
Beide Ausgabe-Pins in den 4 möglichen Kombinationen in einer Endlosschleife ständig durchschalten

Erste Punkte zeichnen (MCU-Programm)




```
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "
#include "

#define XY
#define XY
#define PI
#define PI
#define DE

int main()
{
    SysCtl
    GPIOPi
    while
        GP
        Sy
        GP
        Sy
        GP
        Sy
        GP
        Sy
    }
}
```

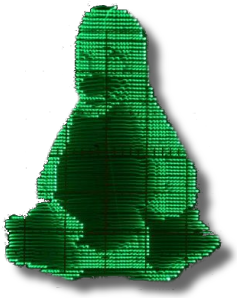
```
                                    // 0,0
                                    // 1,0
                                 ); // 1,1
                                    // 0,1
```



Digital → Analog?

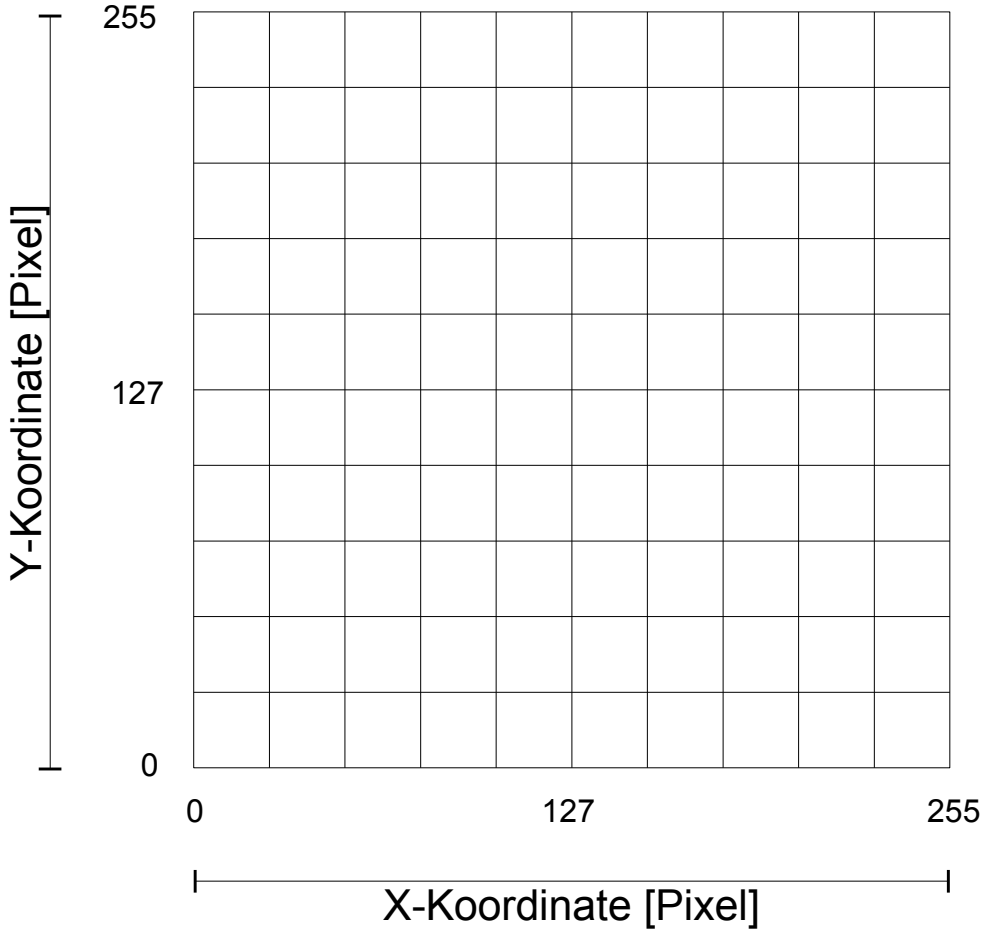
Problem:

- Rechenalgorithmen Computergrafik (Punkte, Linien, Kreise berechnen; Transformation; Skalierung etc.)
  → **digitales** Koordinatensystem
- Ausgabemedium „Kathodenstrahlröhre“
  → **analoge** Spannungen

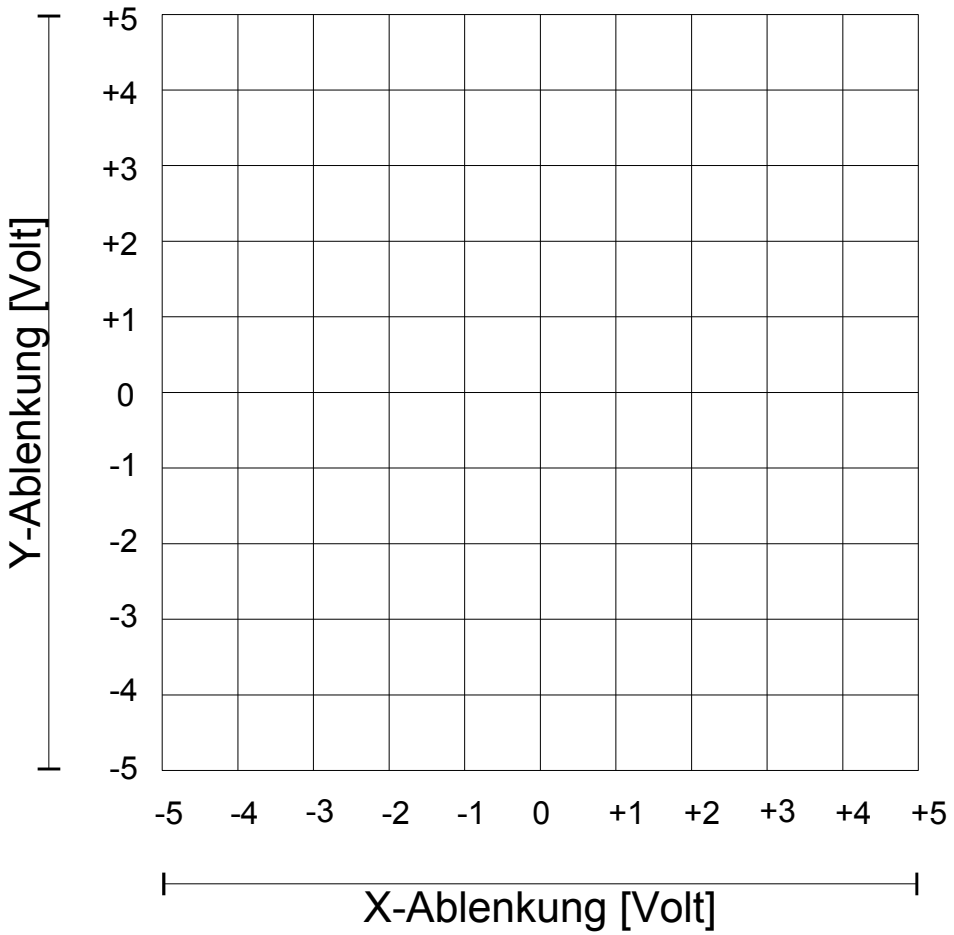


Digital → Analog?

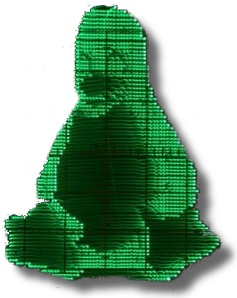
Computer → digital



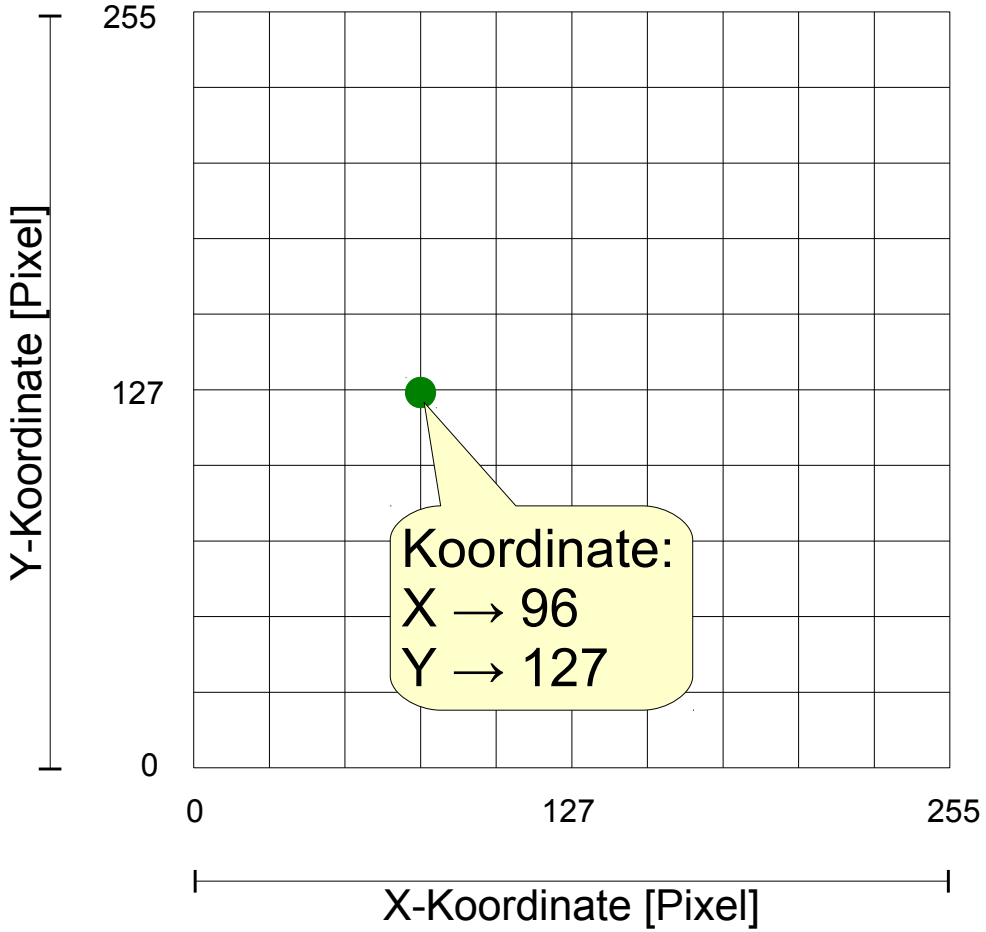
Kathodenstrahlröhre → analog



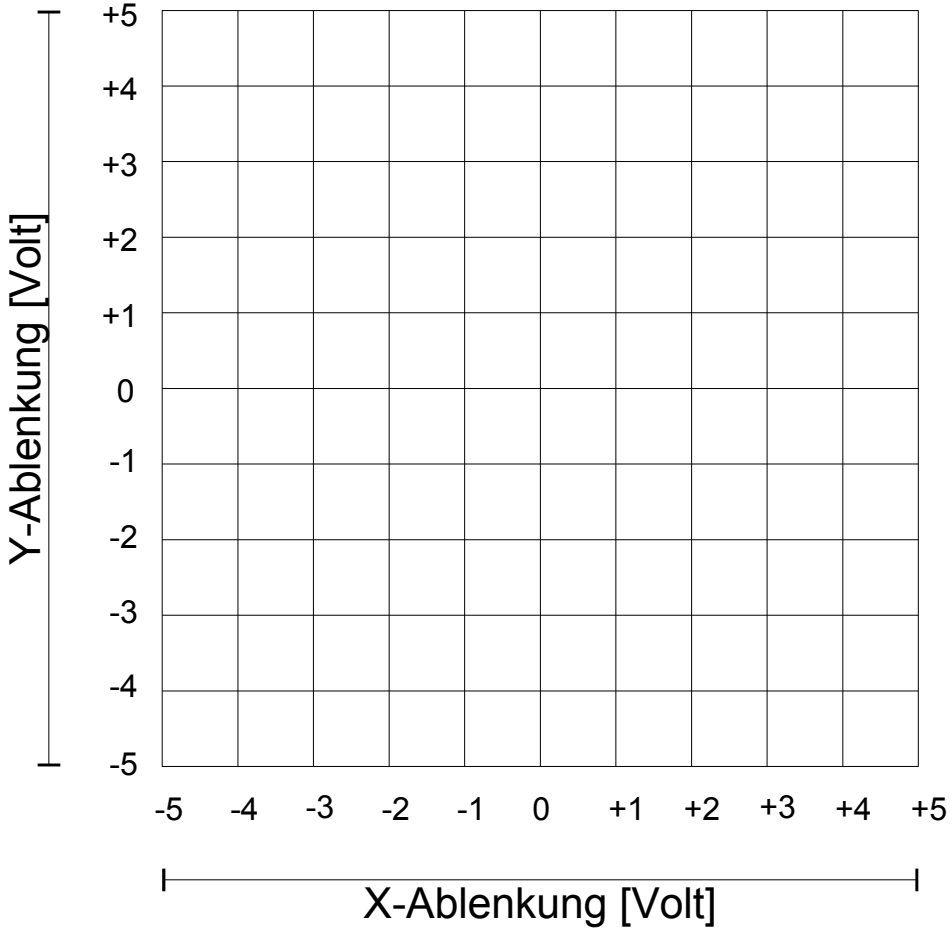
Digital → Analog?

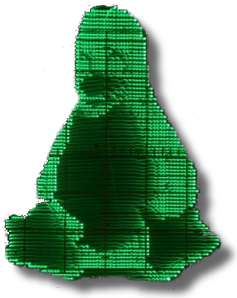


Computer → digital



Kathodenstrahlröhre → analog

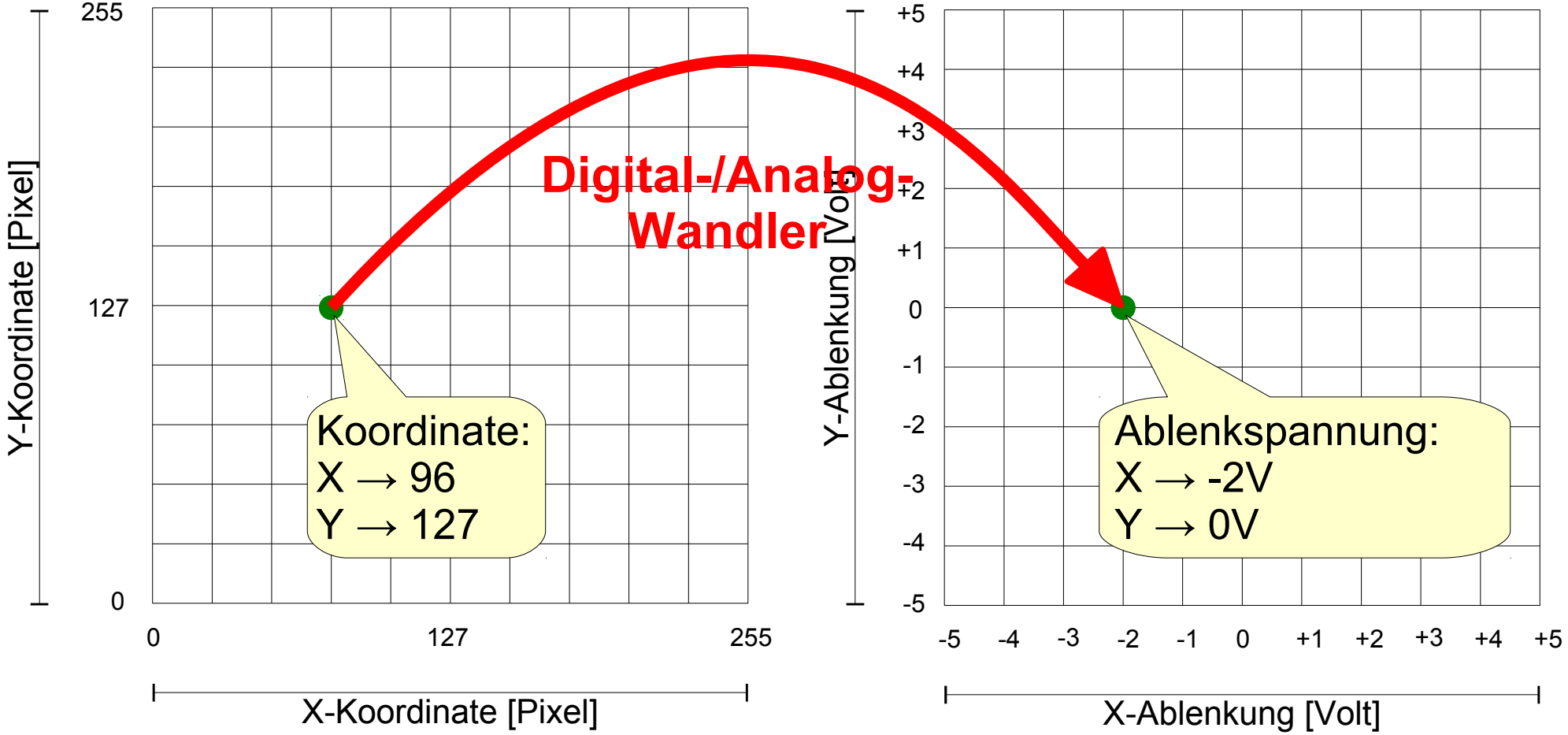




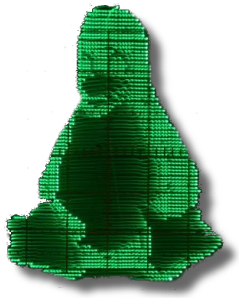
Digital-/Analog-Wandler

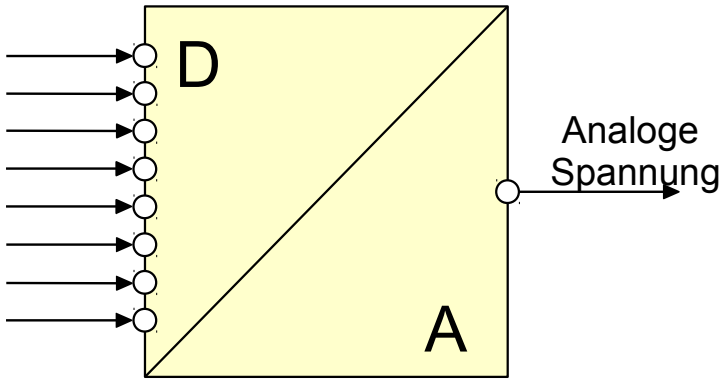
Computer → digital

Kathodenstrahlröhre → analog

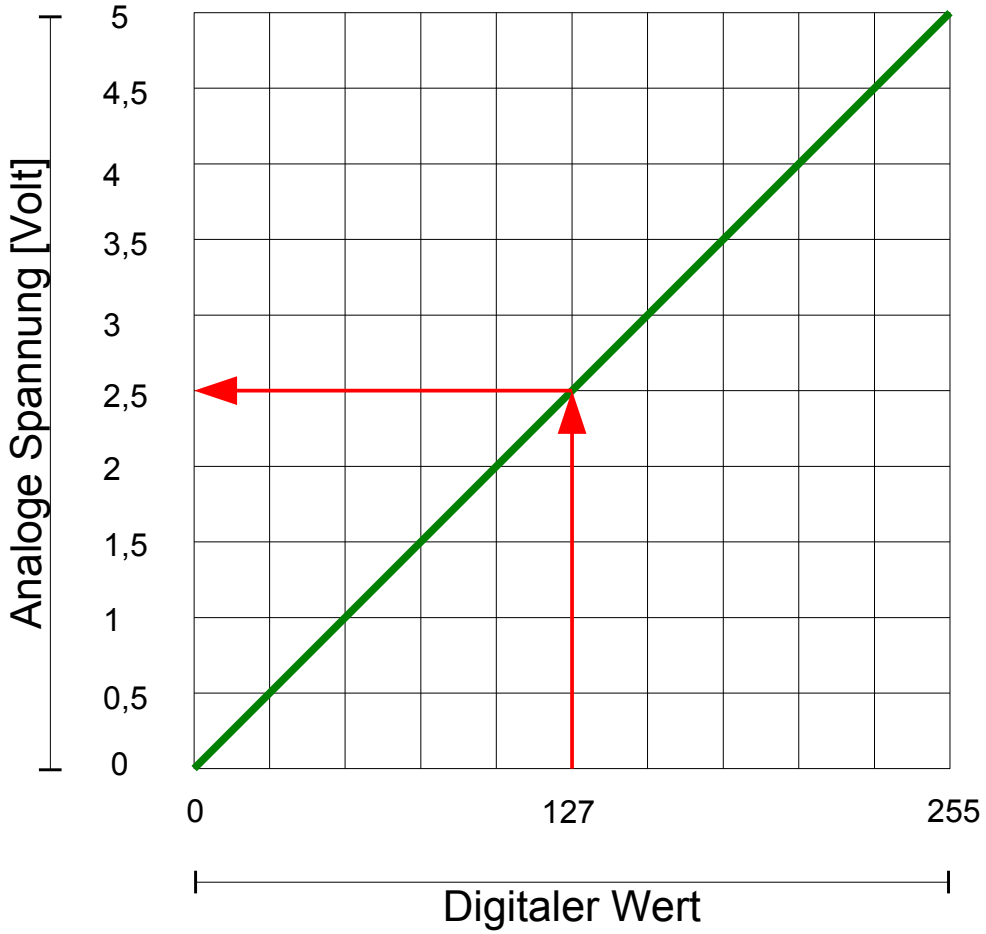


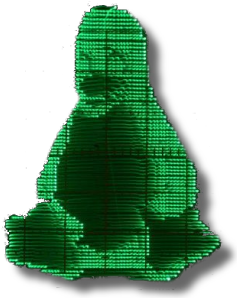
Digital-/Analog-Wandler





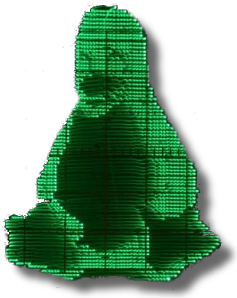
7 D.. 0 Dtr e W ⊕ ati ġ D





Linien zeichnen?

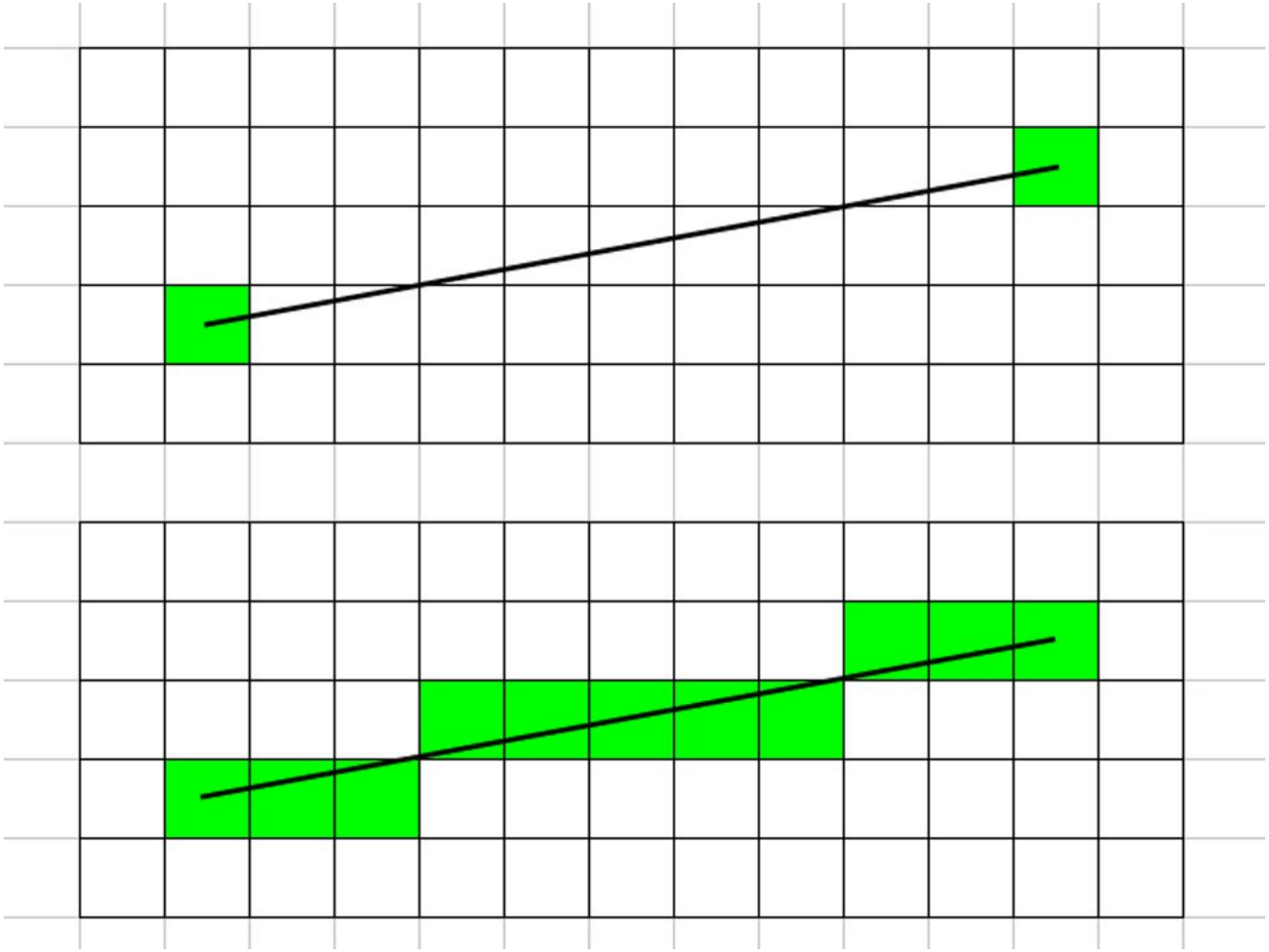
- Wer „Punkte kann“, kann auch Linien, Zeichen etc. darstellen...
- Linien sind durch Anfangs- und Endpunkt im Koordinatensystem definiert
- Dazwischen liegen weitere Punkte, die berechnet und ausgegeben werden müssen:
  - Bresenham-Algorithmus
    → http://de.wikipedia.org/wiki/Bresenham-Algorithmus

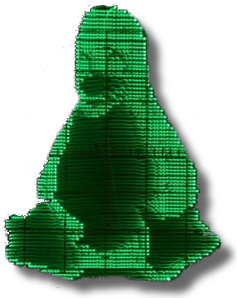


Linien zeichnen?

Anfangs- und Endpunkt sind bekannt

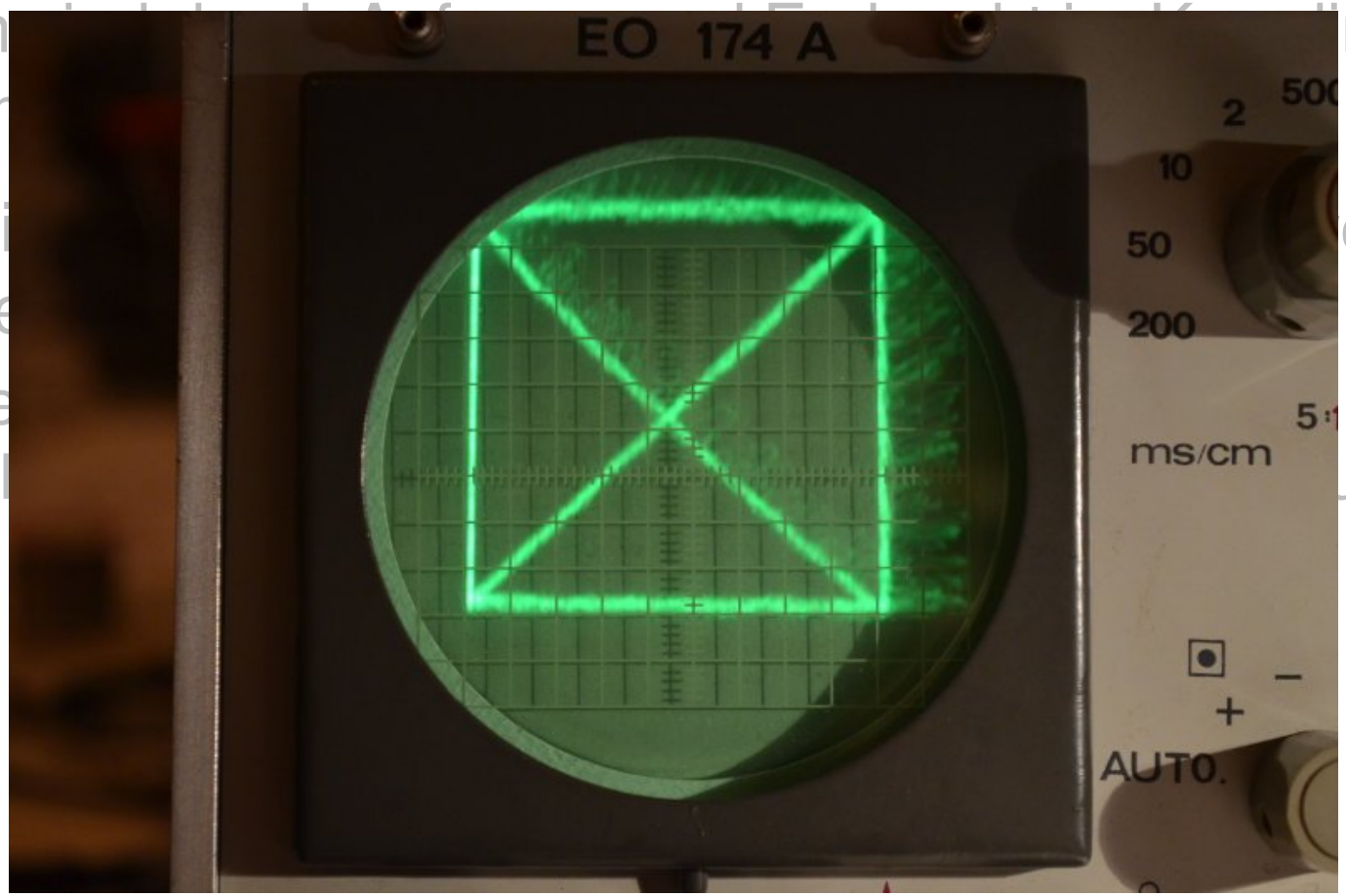
Zwischenpunkte werden berech-net (z.B. nach Bresenham)

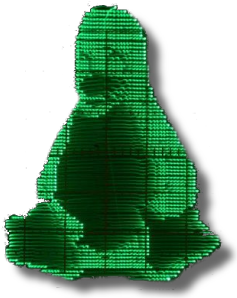




Linien zeichnen?

- Linien
- Dazwi
  - Bre
    → I

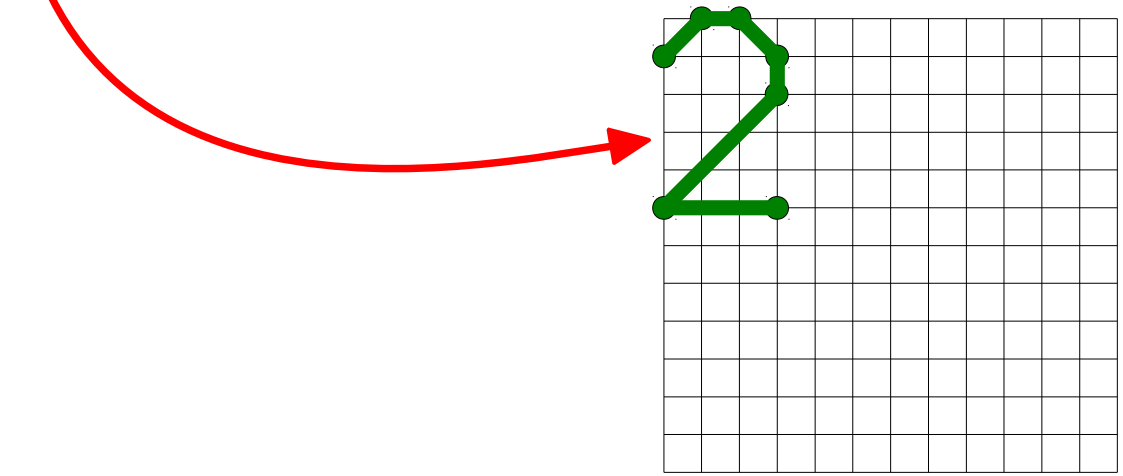




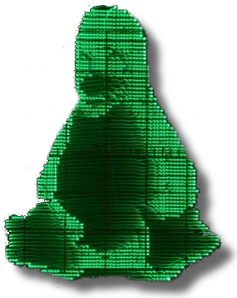
Buchstaben/Zahlen/Zeichen?

- Buchstaben, Zahlen, Zeichen können durch Punkte beschrieben werden, zwischen denen Linien zu zeichnen sind:

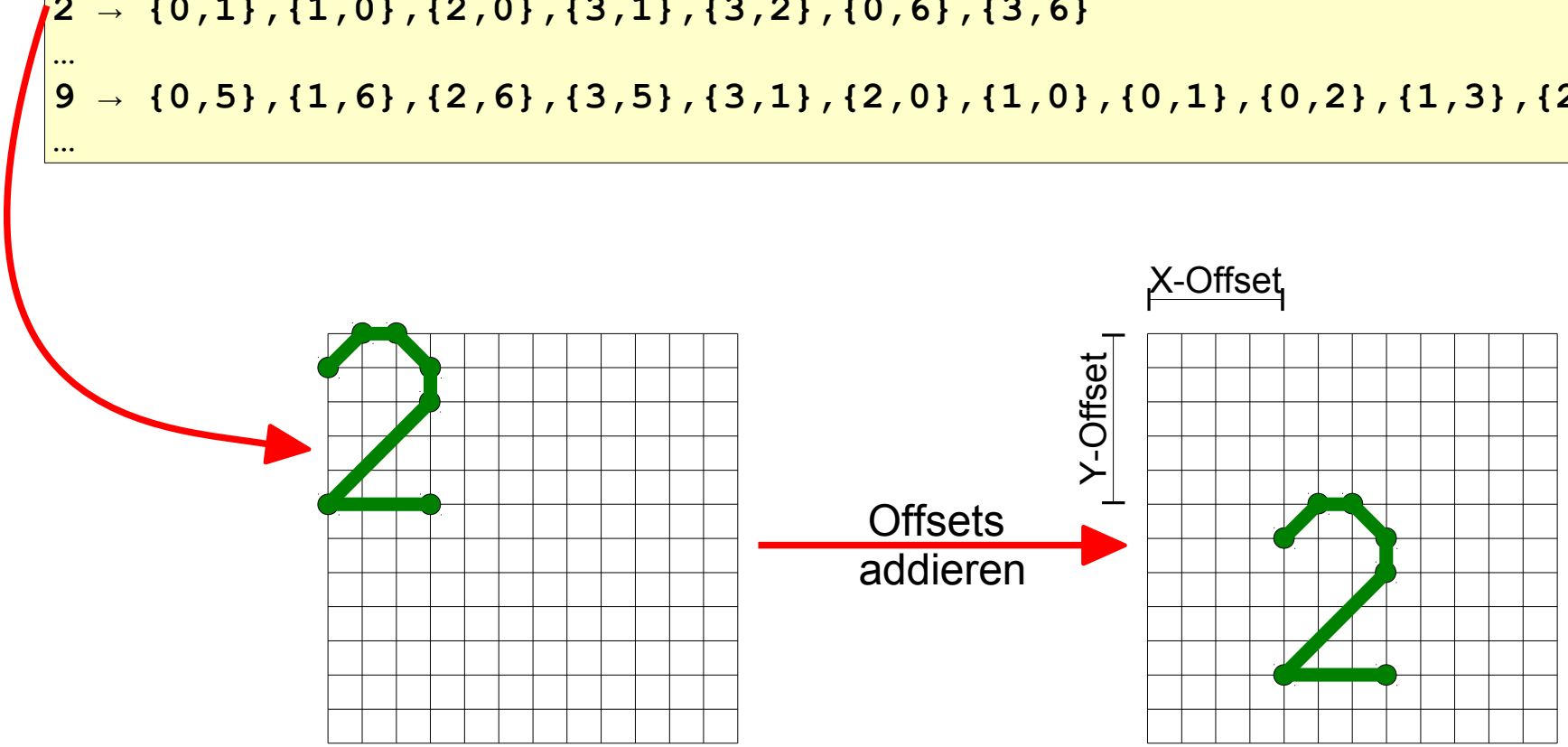
```
0 → {0,5},{0,1},{1,0},{2,0},{3,1},{3,5},{2,6},{1,6},{0,5},{3,1}
1 → {0,2},{2,0},{2,6}
2 → {0,1},{1,0},{2,0},{3,1},{3,2},{0,6},{3,6}
…
9 → {0,5},{1,6},{2,6},{3,5},{3,1},{2,0},{1,0},{0,1},{0,2},{1,3},{2,3},{3,2}
…
```

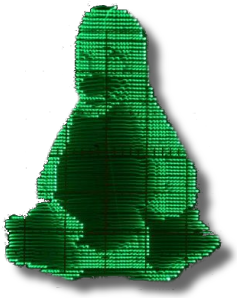


Zeichen auf Bildschirm positionieren?



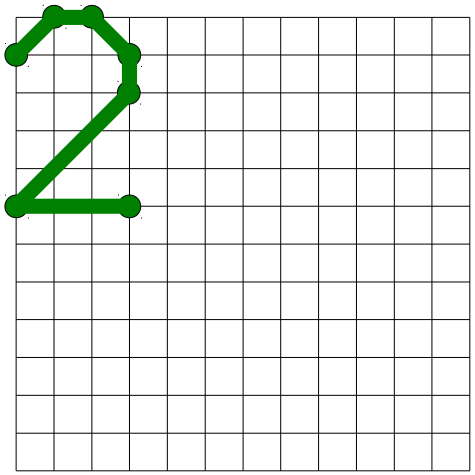
```
0 → {0,5},{0,1},{1,0},{2,0},{3,1},{3,5},{2,6},{1,6},{0,5},{3,1}
1 → {0,2},{2,0},{2,6}
2 → {0,1},{1,0},{2,0},{3,1},{3,2},{0,6},{3,6}
…
9 → {0,5},{1,6},{2,6},{3,5},{3,1},{2,0},{1,0},{0,1},{0,2},{1,3},{2,3},{3,2}
…
```



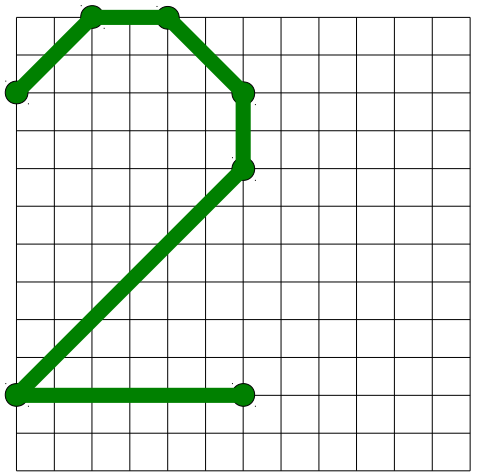


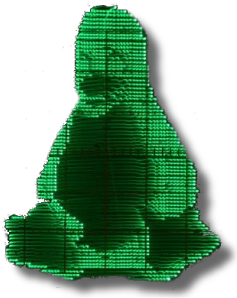
Zeichen vergrößern?

```
0 → {0,5},{0,1},{1,0},{2,0},{3,1},{3,5},{2,6},{1,6},{0,5},{3,1}
1 → {0,2},{2,0},{2,6}
2 → {0,1},{1,0},{2,0},{3,1},{3,2},{0,6},{3,6}
…
9 → {0,5},{1,6},{2,6},{3,5},{3,1},{2,0},{1,0},{0,1},{0,2},{1,3},{2,3},{3,2}
…
```



mit Vergrößerungs-
faktor multiplizieren





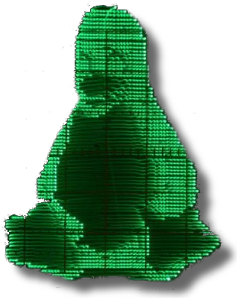
Punkte eines Kreises zeichnen?

Problem:

- Es werden Winkelfunktionen (Sinus, Cosinus) benötigt, die viel Rechenleistung verbrauchen
- MCUs „rechnen nicht gern“ mit gebrochenen Zahlen...

Mögliche Lösungen:

- Auch hier hat Bresenham einen Algorithmus...
- Vorberechnete Festkommawerte für sin(x), cos(x) in Lookup-Tabellen ablegen
- Näherungsformeln/-reihen für Winkelfunktionen verwenden



Punkte eines Kreises zeichnen?

Meine Wahl:

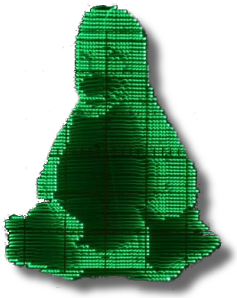
- Lookup-Tabelle für Sinus und Cosinus:
  - Jeweils für einen 90°-Kreisbogen → Kreissymmetrie...
  - Jeweils nur 15 Werte → 15 * 4 = 60 Minuten...

```
// Sinusreihe --> sin(x*2*pi/60)*1000 mit x=0...15

static const int tab_sin[] =
{0,105,208,309,407,500,588,669,743,809,866,914,951,978,995,1000};



// Cosinusreihe --> cos(x*2*pi/60)*1000 mit x=0...15

static const int tab_cos[] =
{1000,995,978,951,914,866,809,743,669,588,500,407,309,208,105,0};
```

Simulator


ScopeClock
12
1
2
3
4
5
6
7
8
9
10
11
Mode
Zoom: 3
sdx: 3
sdy: 10
Digits:


ScopeClock
03.05.12
15:31:30
Mode
Zoom: 3
sdx: 3
sdy: 10
Digits:



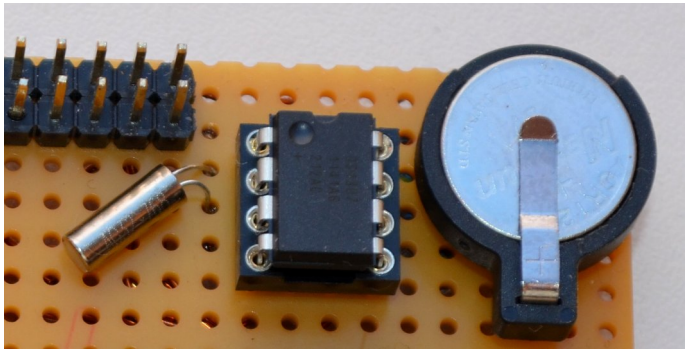
Genauer Sekundentakt...

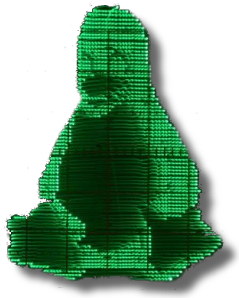
Problem:

- Generierung eines möglichst genauen Sekundentakts
- Weiterzählen der Uhrzeit im ausgeschalteten Zustand

Lösung:

- z.B. einen batteriegepufferten RTC-Schaltkreis DS1307 mit „Uhrenquarz“ verwenden:
  - Uhrenquarz → 32,768kHz
  - Pufferbatterie → CR1225 (3V)
  - Kommunikation via I^2C-Bus
  - Sekundentakt-Ausgang vorhanden

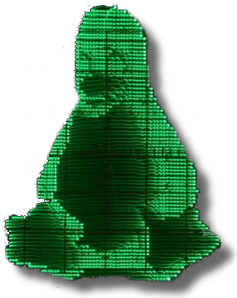




Grenzen suchen und finden...

Wie schnell muss eigentlich der verwendete Mikrocontroller (MCU) sein, um:

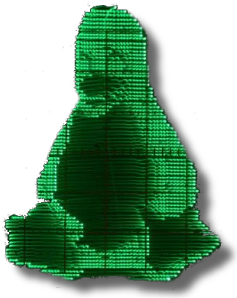
- Die Uhrzeit, Taster etc. zu verwalten?
  → vernachlässigbar...
- Das auszugebende Bild „ausreichend schnell“ zu berechnen?
- Alle Bildpunkte „ausreichend schnell“ auszugeben?



Grenzen suchen und finden...

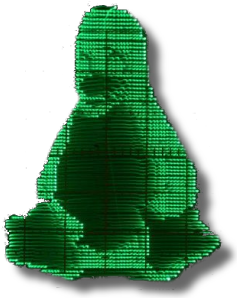
Was heißt „ausreichend schnell“?

- Flimmerfreies Bild → mindestens 50 Bilder pro Sekunde
  - 1s / 50 = 20ms pro Bild
  - 1 Bild enthält z.B. 2000 Punkte
    → 20ms / 2000 = **0,01ms pro Bildpunkt**
- ...das bedeutet im Idealfall („Milchmädchenrechnung“):
  - 20-MHz-MCU → 200 Maschinenbefehle pro Bildpunkt
  - 80-MHz-MCU → 800 Maschinenbefehle pro Bildpunkt



Alternativen betrachten...

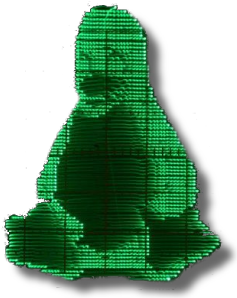
- Bildschirmpuffer verwenden:
  - Vorteil: Bildpunkte müssen nur dann berechnet werden, wenn sich etwas an der Anzeige ändert
  - Nachteil: Bildschirmpuffer muss immer vollständig ausgelesen/-gegeben werden (256 x 256 = 65536 Pixel...!)
- Jeden Bildpunkt sofort nach seiner Berechnung ausgeben:
  - Vorteil: es werden nur soviel Bildpunkte ausgegeben, wie notwendig
  - Nachteil: jeder Bildpunkt muss zur Ausgabe eines Bildes neu berechnet werden (also mind. 50x in der Sekunde...)



Lösungen finden...

Realisierte Lösung:

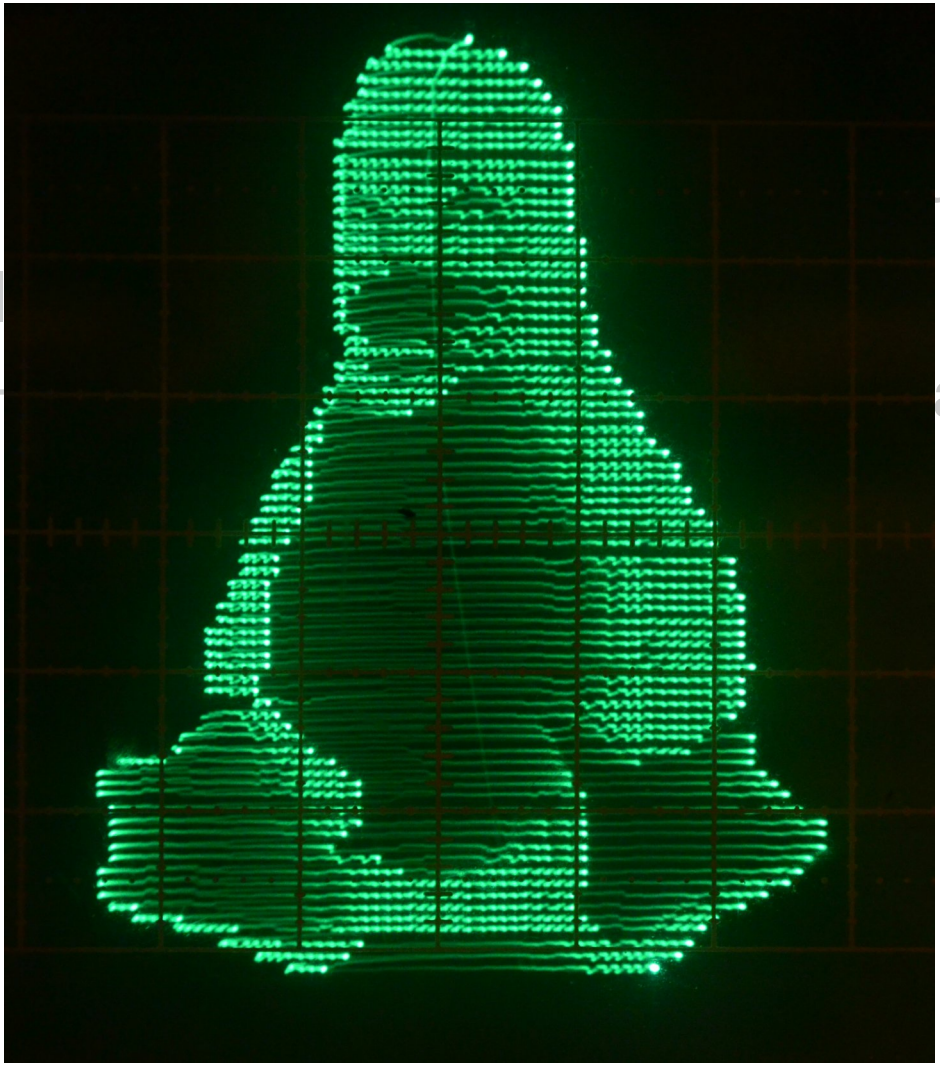
- Jeden Bildpunkt sofort nach seiner Berechnung ausgeben
- Einen schnellen Mikrocontroller verwenden:
  - LM4F120H5QR auf einem Stellaris Launchpad...

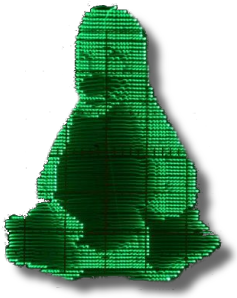


Lösungen finden...

Meine Lösung:

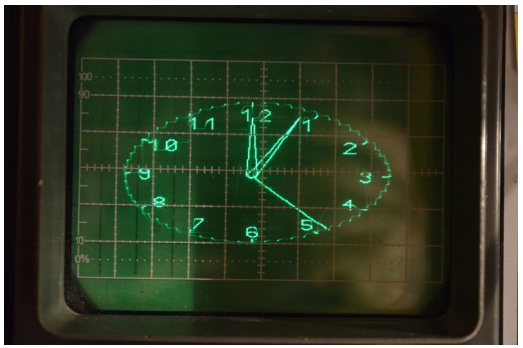
- Jeden Bildpu ung ausgeben
- Einen schnell
  - LM4F120H ad...

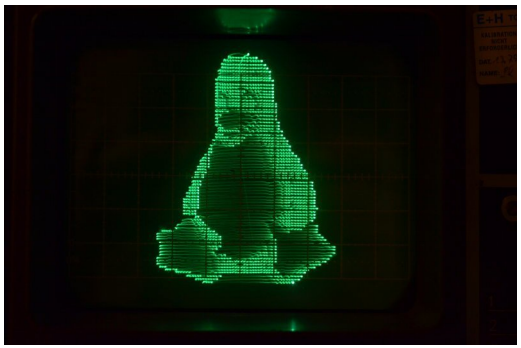


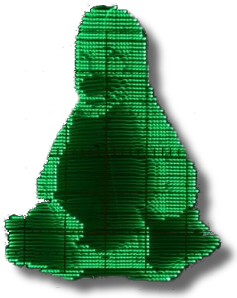


...Lösungen auch mal „überprüfen“

CPU-Takt [Mhz]	Anzeige	Zeit pro Bild [ms]	Pixel pro Bild	Bildwieder-holfrequenz [Hz]	Zeit pro Pixel [ms]
80	Digitaluhr	2,6	1180	**384,6**	0,00220
	Analoguhr	2,85	1250	**350,9**	0,00228
	Tux	3,5	1356	**285,7**	0,00258
	Vollbild 255x255	123,5	65025	**8,1**	0,00190
	Vollbild 128x128	31,1	16384	**32,2**	0,00190

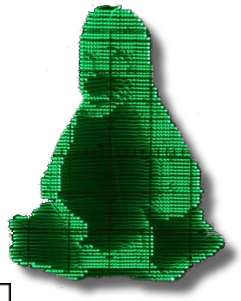




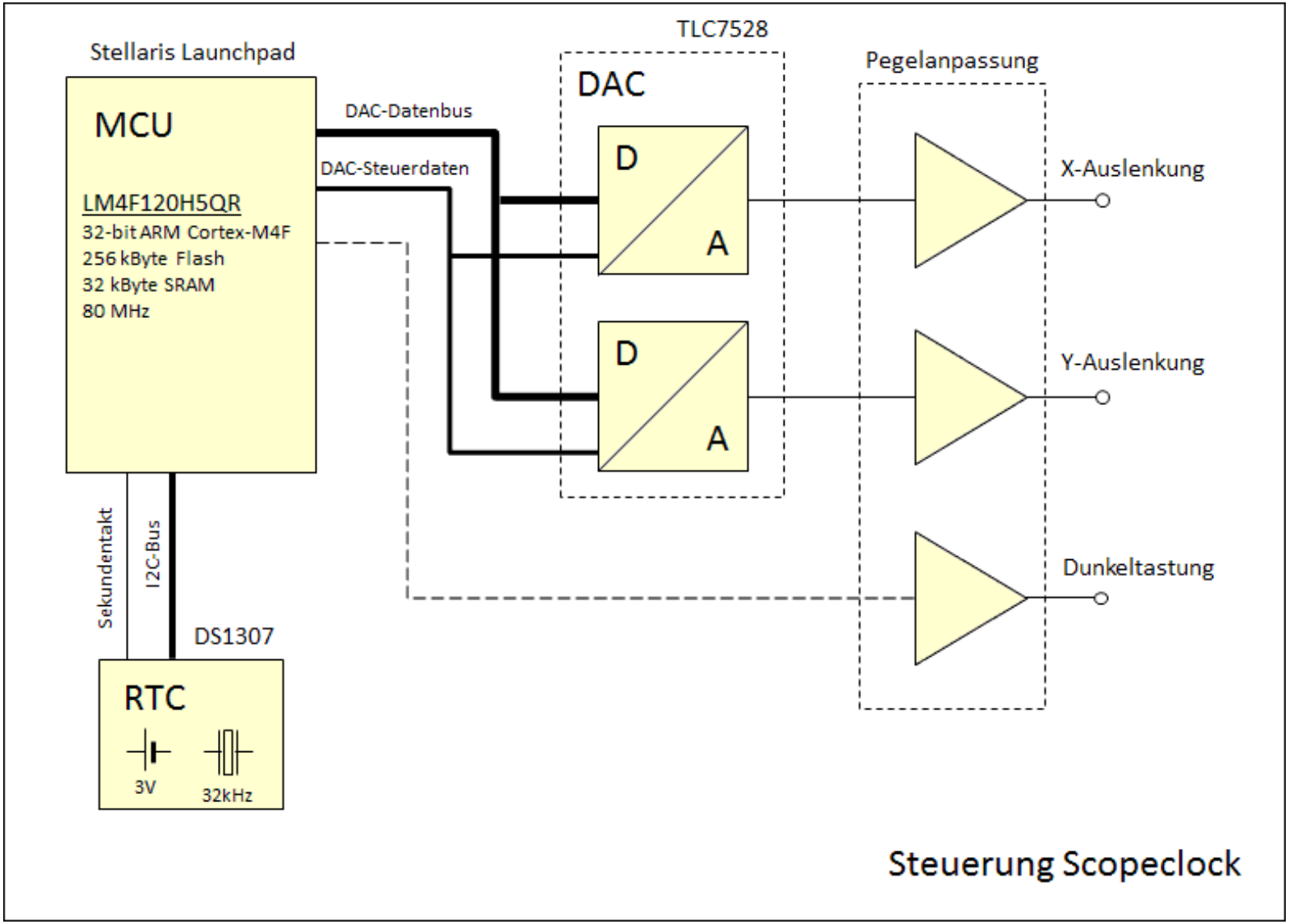


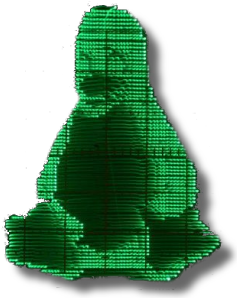
Hardware aufbauen

- Schaltplan entwerfen
  - sich an seine Elektronikvorlesungen (o.ä.) erinnern
  - Datenblätter wälzen
- Stückliste erstellen und einkaufen gehen
- Testaufbau (Breadboard, Lochraster)
- eventuell Leiterplatte entwerfen und herstellen
- Lötkolben anheizen („Beruhigungsbier“ trinken, Lesebrille suchen...)
- Lötergebnis ausgiebig(!) kontrollieren



Schaltplan entwerfen

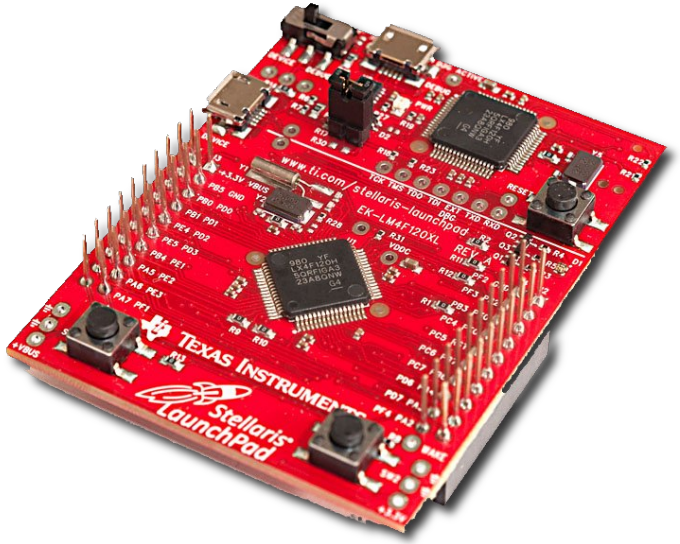


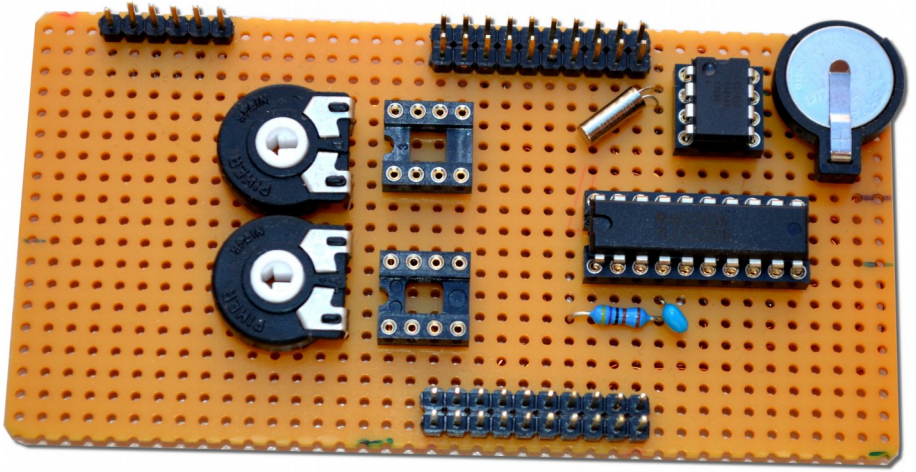


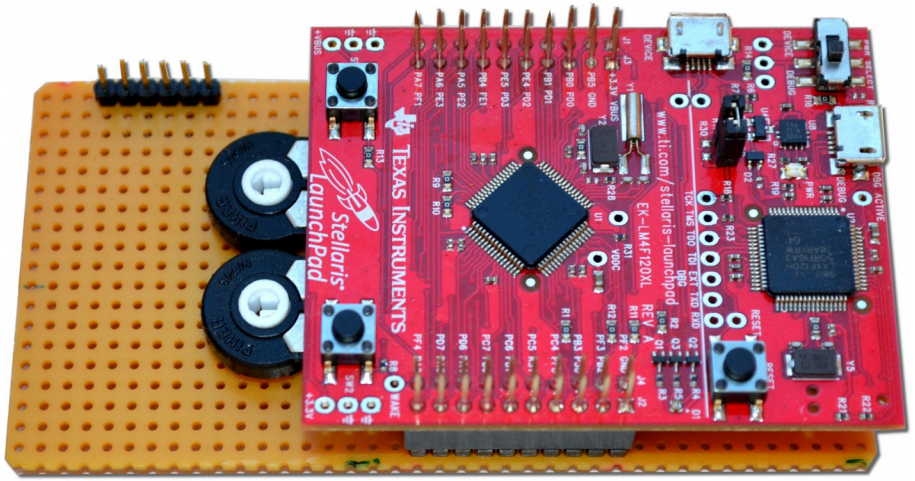
Auswahl Digital-Hardware

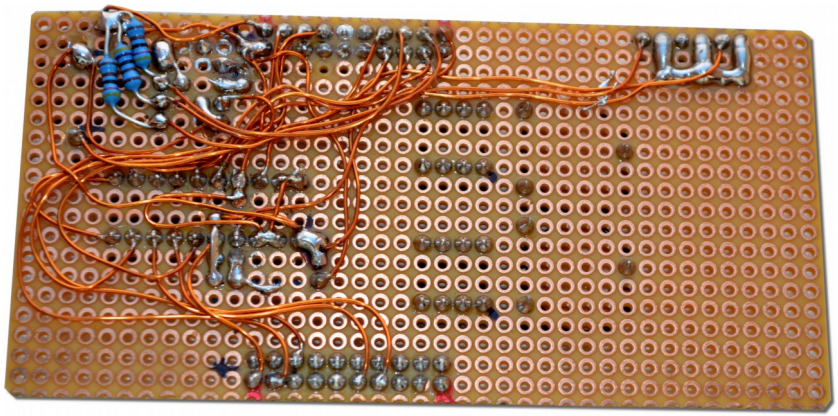
- Mikrocontroller:
  - LM4F120H5QR auf Stellaris Launchpad von TI
    - 32-Bit ARM Cortex-M4F
    - 80 MHz CPU-Takt
    - 256kByte Flash, 32kByte RAM
- Digital-/Analog-Wandler (DAC):
  - TLC7528:
    - 2 Kanäle (X-, Y-Auslenkung...)
    - 8 Bit Auflösung
- RTC:
  - DS1307 mit Uhrenquarz und Pufferbatterie
    - Anbindung via I^2C-Bus

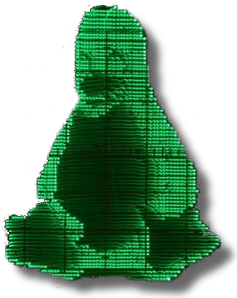
Hardware aufbauen



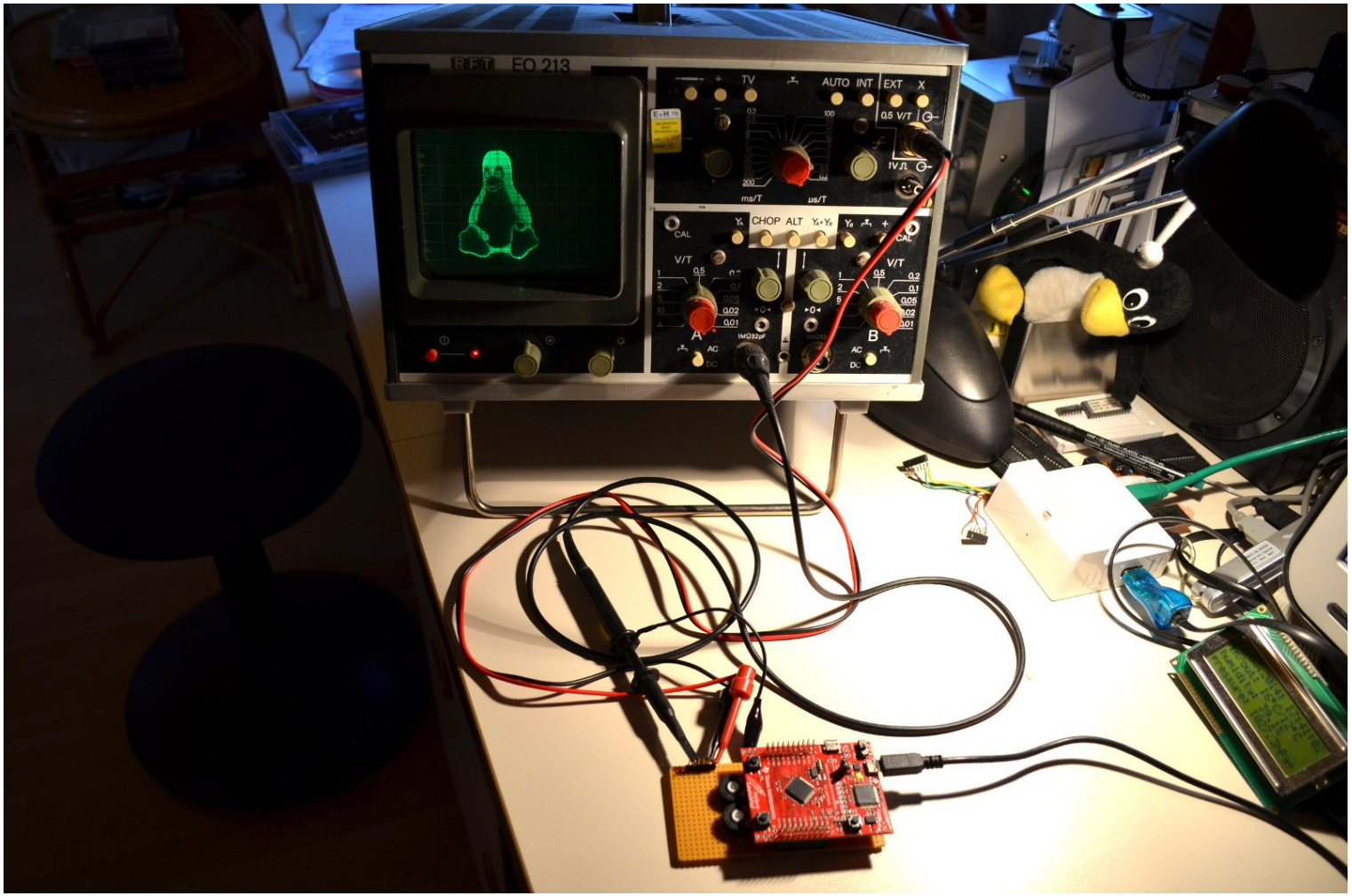


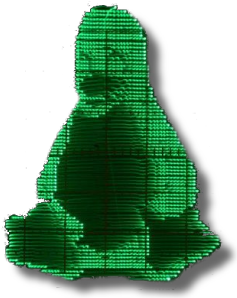






Hardware aufbauen



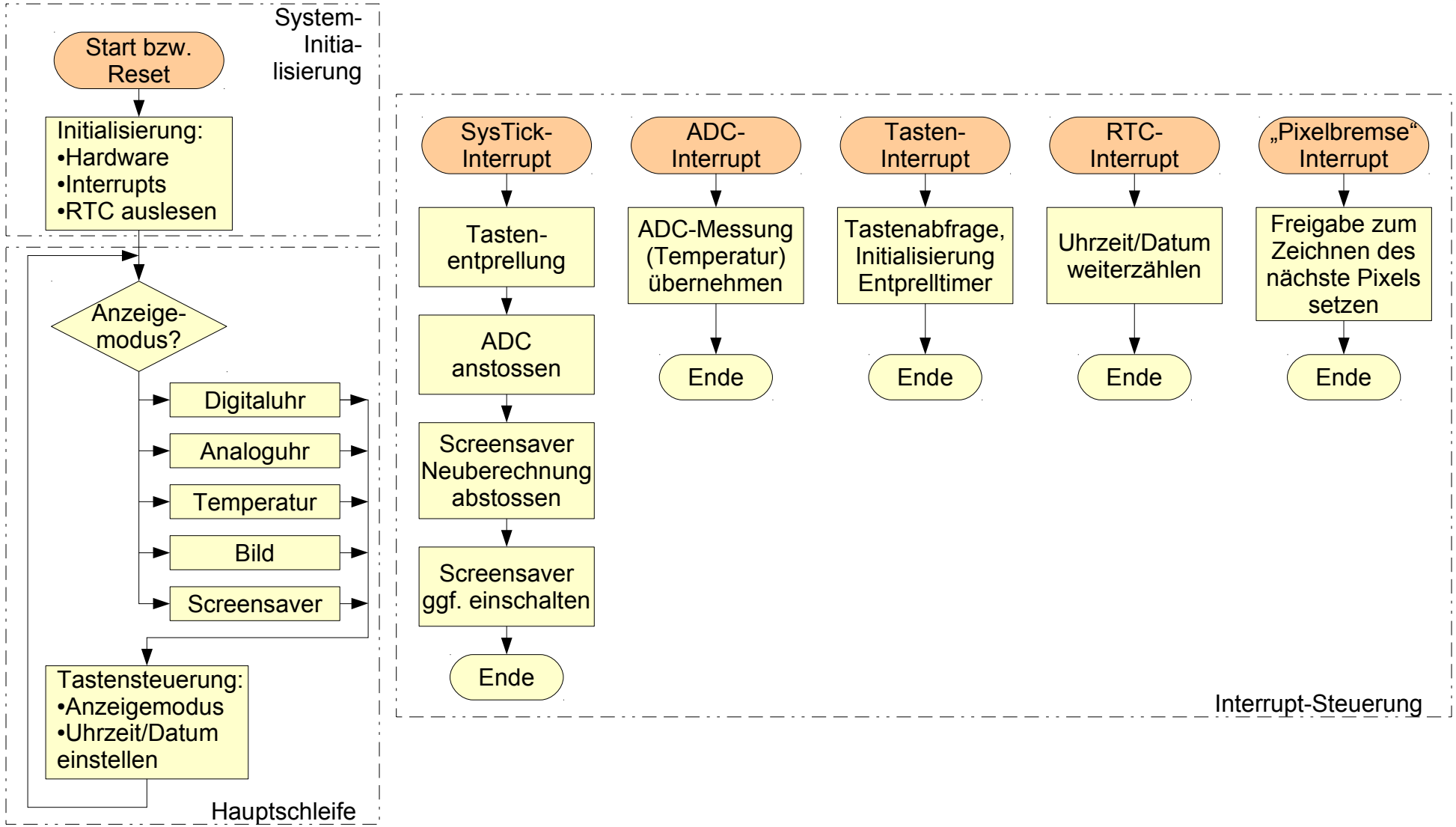


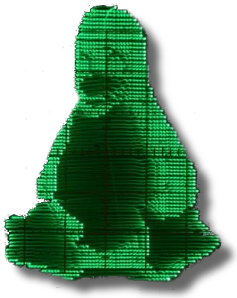
MCU-Firmware entwickeln (Toolchain)

- Gewählte Programmiersprache → C
- Toolchain für Stellaris Launchpad:
  - arm-none-eabi-gcc
  - binutils
  - StellarisWare (http://www.ti.com/tool/sw-ek-lm4f120xl)
  - lm4flash (Flash-Tool)
  - Ein Editor/IDE...
- Toolchain mal mit einem „Hello World“ ausprobieren

- Tipp für „Warmduscher“:
  → Energia (http://energia.nu/)

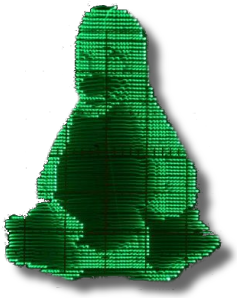
MCU-Firmware entwickeln (PAP)





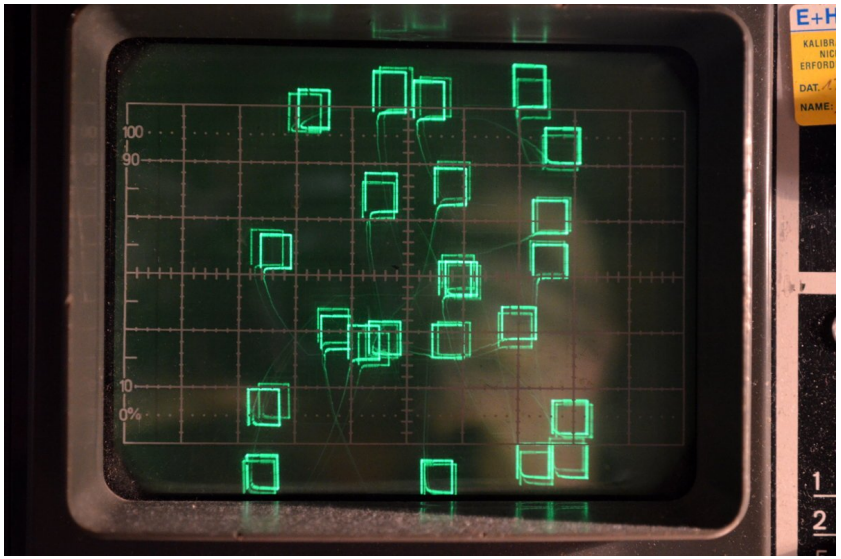
Test, Debugging

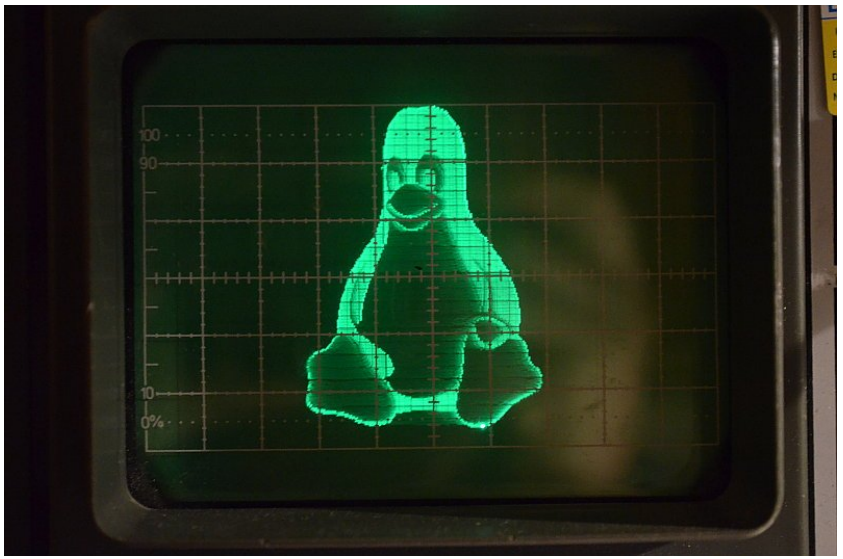
- Programm auf MCU laden, einschalten und geht...?
  → ...schön wäre es!
- Hard- und Software schrittweise testen und in Betrieb nehmen!
- Es muss nicht immer ein Debugger sein!
  - Logikanalysator, Oszilloskop ;-), Multimeter etc.
  - LEDs
  - serielle Schnittstelle

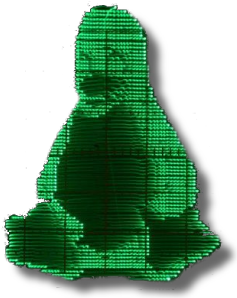


Immer noch Langeweile...?

- Ein paar Gimmicks für meine Scopeclock:
  - „Bildschirmschoner“
    → fliegende Quadrate, die am „Rand“ abprallen
  - Bilder anzeigen
    → Darstellung von s/w-Bildern im xbm-Format (X-Bitmap; http://en.wikipedia.org/wiki/X_BitMap)

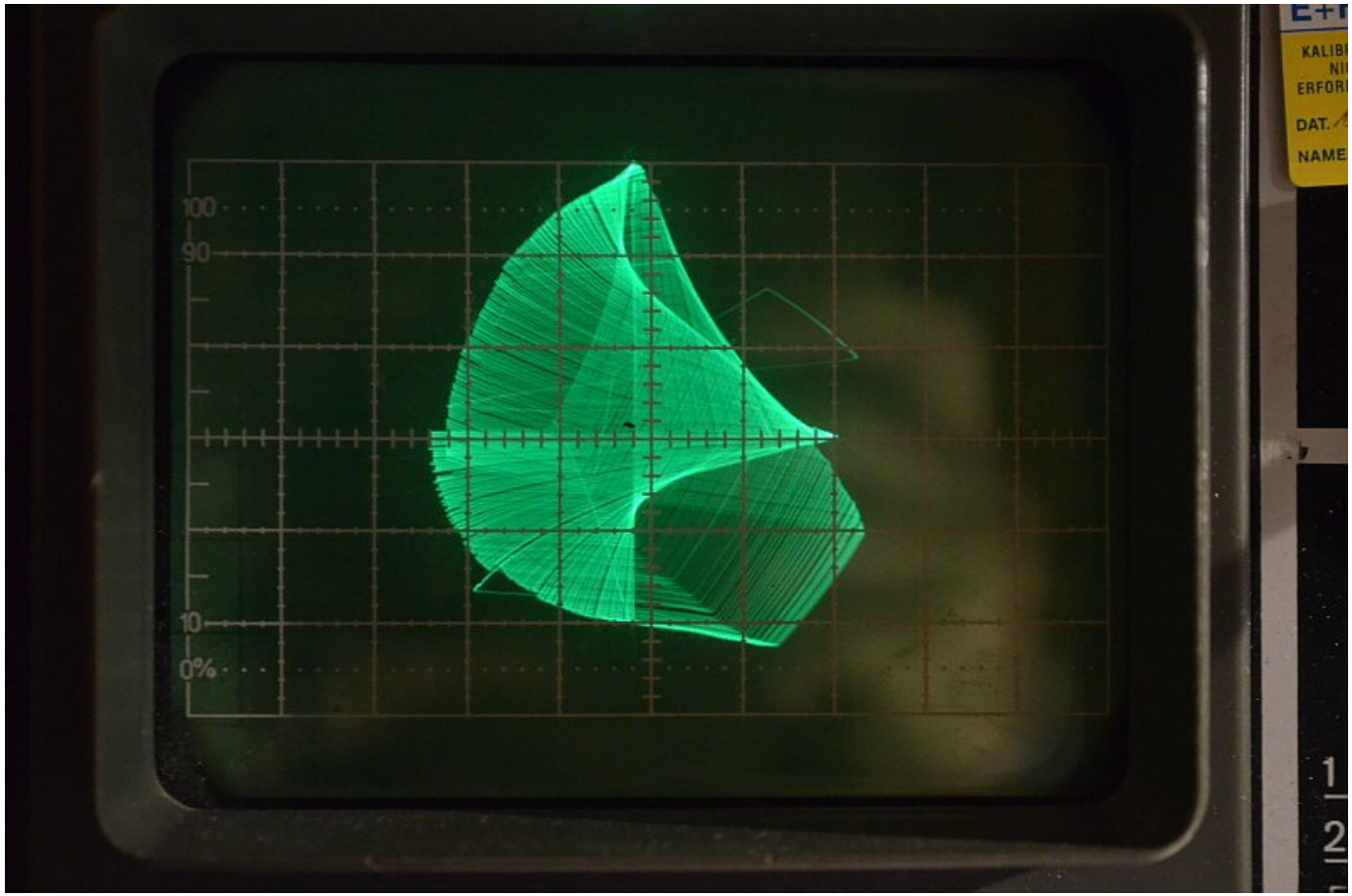


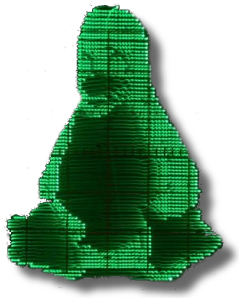




Ein kleines Rätsel...

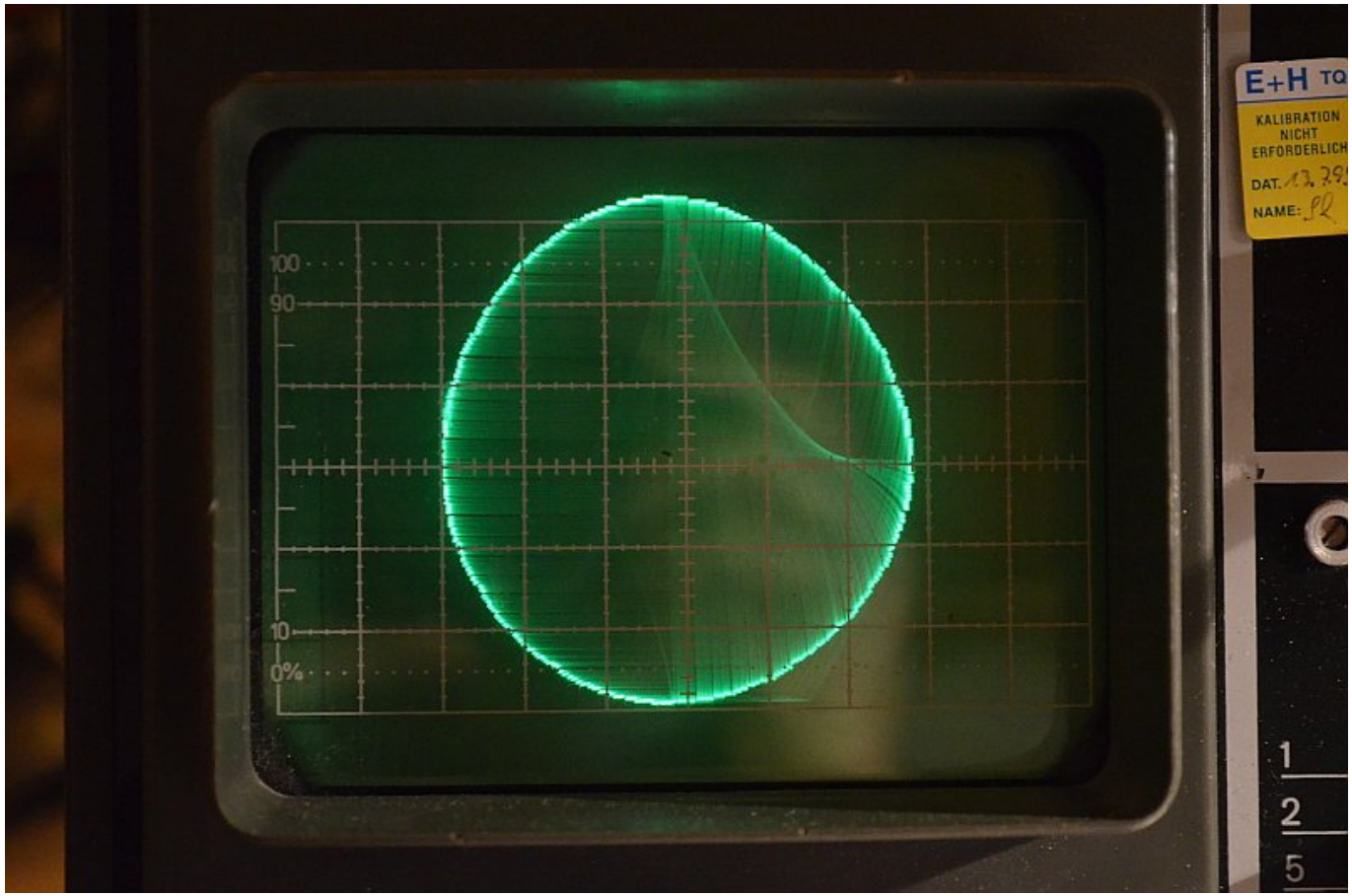
Was ist das?



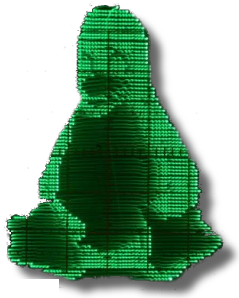


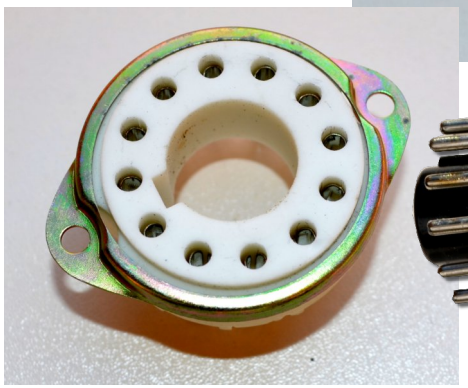
Ein kleines Rätsel...

...ein Kreis mit dem Bresenham-Algorithmus berechnet!



Was fehlt derzeit noch...?





PHILIPS DG7-32

Limiting values (design center values)
Caractéristiques limites (valeurs moyennes d'étude)
Grenzdaten (mittlere Entwicklungsdaten)

$V_{g2,g4}$	= max.	800	V
$V_{g2,g4}$	= min.	400	V
V_{g3}	= max.	200	V[4])
$-V_{g1}$	= max.	160	V
$+V_{g1}$	= max.	0	V
$V_{D1D1'p}$	= max.	450	V
$V_{D2D2'p}$	= max.	750	V
V_{kf}	= max.	125	V
W_l	= max.	3	mW/cm²
W_{g2+g4}	= max.	0,5	W

Max. circuit values
Valeurs max. des éléments du montage
Max. Werte der Schaltungsteile

R_D	= max.	5	MΩ
R_{g1}	= max.	0,5	MΩ

4.

tentiometer a grid 3
+10 µA must be taken

la grille 3 il faut
ille 3 de -15 µA au

von Gitter 3 muss
x. +10 µA Rechnung

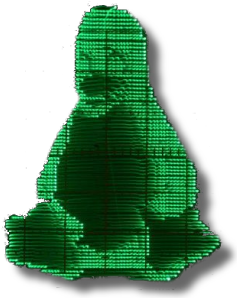
l extinction of the

inction visuelle du

ptische Löschung des

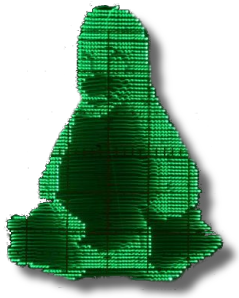
fokussierten

12.12.1957 938 2730 3.



Weiterführende Informationen

- http://www.bralug.de/wiki/Scopeclock
- http://bralug.de/wiki/Tux_fliegt_zu_den_Sternen



Danke für die Aufmerksamkeit!