Raspberry Pi GPIO Pin Hacking

Bernhard Trummer
bernhard.trummer@gmx.at
PGP key-id: 0385D2E2

4. April 2014

Über mich

- TU / Telematik (1994 – 2001)
- Linux User Group Graz
- Angestellt bei Infonova (seit 2000)

Inhalt

- Raspberry Pi und GPIO (>>)
- wiringPi (>>)
- Beispiel: Ampelsteuerung per Shell-Script (>>)
- Crash-Kurs: Taster und LEDs (>>)
- Beispiel: KITT meets BSG
- Crash-Kurs: Schieberegister und Latches
- Beispiel: Helligkeitssteuerung / PWM
- Q & A

Inhalt

- Raspberry Pi und GPIO (>>)
- wiringPi (>>)
- Beispiel: Ampelsteuerung per Shell-Script (>>)
- Crash-Kurs: Taster und LEDs (>>)
- Beispiel: KITT meets BSG
- Crash-Kurs: Schieberegister und Latches
- Beispiel: Helligkeitssteuerung / PWM
- Q & A

Inhalt

- Raspberry Pi und GPIO (>>)
- wiringPi (>>)
- Beispiel: Ampelsteuerung per Shell-Script (>>)
- Crash-Kurs: Taster und LEDs (>>)
- Beispiel: KITT meets BSG
- Crash-Kurs: Schieberegister und Latches
- Beispiel: Helligkeitssteuerung / PWM
- Q & A

Inhalt

- Raspberry Pi und GPIO (>>)
- wiringPi (>>)
- Beispiel: Ampelsteuerung per Shell-Script (>>)
- Crash-Kurs: Taster und LEDs (>>)
- Beispiel: KITT meets BSG
- Crash-Kurs: Schieberegister und Latches
- Beispiel: Helligkeitssteuerung / PWM
- Q & A

Inhalt

- Raspberry Pi und GPIO (>>)
- wiringPi (>>)
- Beispiel: Ampelsteuerung per Shell-Script (>>)
- Crash-Kurs: Taster und LEDs (>>)
- Beispiel: KITT meets BSG
- Crash-Kurs: Schieberegister und Latches
- Beispiel: Helligkeitssteuerung / PWM
- Q & A

Inhalt

- Raspberry Pi und GPIO (>>)
- wiringPi (>>)
- Beispiel: Ampelsteuerung per Shell-Script (>>)
- Crash-Kurs: Taster und LEDs (>>)
- Beispiel: KITT meets BSG
- Crash-Kurs: Schieberegister und Latches
- Beispiel: Helligkeitssteuerung / PWM
- Q & A

Inhalt

- Raspberry Pi und GPIO (>>)
- wiringPi (>>)
- Beispiel: Ampelsteuerung per Shell-Script (>>)
- Crash-Kurs: Taster und LEDs (>>)
- Beispiel: KITT meets BSG
- Crash-Kurs: Schieberegister und Latches
- Beispiel: Helligkeitssteuerung / PWM
- Q & A

Inhalt

- Raspberry Pi und GPIO (>>)
- wiringPi (>>)
- Beispiel: Ampelsteuerung per Shell-Script (>>)
- Crash-Kurs: Taster und LEDs (>>)
- Beispiel: KITT meets BSG
- Crash-Kurs: Schieberegister und Latches
- Beispiel: Helligkeitssteuerung / PWM
- Q & A

Raspberry Pi und GPIO

Raspberry Pi (>>)

- `http://www.raspberrypi.org/`
- credit-card sized computer
- 1x Ethernet (100 Mbit), 2x USB (Model B)
- HDMI, Composite- und Kopfhörerausgang
- 17 (Model A) bzw. 21 (Mobel B) GPIO Pins

GPIO (>>)

- General Purpose Input/Output
- ein Pin, der als Input oder Output frei programmierbar ist
- leicht zugänglich ist

Spezielle GPIO Features (>>)

- UART
- I2C
- SPI
- Clock (konfigurierbarer Teiler)
- PWM

Anwendungsmöglichkeiten (>>)

- Tasten und LEDs
- LCDs
- Relais, (Schritt)motoren, Servos
- usw.
- General Purpose := Dinge tun, die vom Hersteller nicht vorgesehen sind. ;-)

Anwendungsmöglichkeiten (>>)

- Tasten und LEDs
- LCDs
- Relais, (Schritt)motoren, Servos
- usw.
- General Purpose := Dinge tun, die vom Hersteller nicht vorgesehen sind. ;-)

Raspberry Pi vs. Arduino vs. Gnublin vs. ...

- ARM CPU vs. Mikrocontroller
- Linux + Flexibilität
- vs. low-level, Timing
- alle Beispiele die ich zeigen werde funktionieren prinzipiell auf jedem „Ding“ mit GPIO Pins.

Raspberry Pi vs. Arduino vs. Gnublin vs. ...

- ARM CPU vs. Mikrocontroller
- Linux + Flexibilität
- vs. low-level, Timing
- alle Beispiele die ich zeigen werde funktionieren prinzipiell auf jedem „Ding“ mit GPIO Pins.

Raspberry Pi vs. Arduino vs. Gnublin vs. ...

- ARM CPU vs. Mikrocontroller
- Linux + Flexibilität
- vs. low-level, Timing
- alle Beispiele die ich zeigen werde funktionieren prinzipiell auf jedem „Ding“ mit GPIO Pins.

Raspberry Pi vs. Arduino vs. Gnublin vs. ...

- ARM CPU vs. Mikrocontroller
- Linux + Flexibilität
- vs. low-level, Timing
- alle Beispiele die ich zeigen werde funktionieren prinzipiell auf jedem „Ding“ mit GPIO Pins.

Prototyping (>>)

- Steckplatine (Breadboard)
- Verbindungskabel
- Selbstbau-Kabel vor dem ersten Einsatz unbedingt auf Kurzschlüsse prüfen. :-)

Prototyping (>>)

- Steckplatine (Breadboard)
- Verbindungskabel
- Selbstbau-Kabel vor dem ersten Einsatz unbedingt auf Kurzschlüsse prüfen. :-)

Prototyping (>>)

- Steckplatine (Breadboard)
- Verbindungskabel
- Selbstbau-Kabel vor dem ersten Einsatz unbedingt auf Kurzschlüsse prüfen. :-)

wiringPi (>>)

- `http://wiringpi.com/`
- C Library um GPIO Pins nutzbar zu machen
- angelehnt an Arduino
- gpio Commandline Tool (für Shell Scripts)
- Bindings für Python, PHP, Ruby, Perl

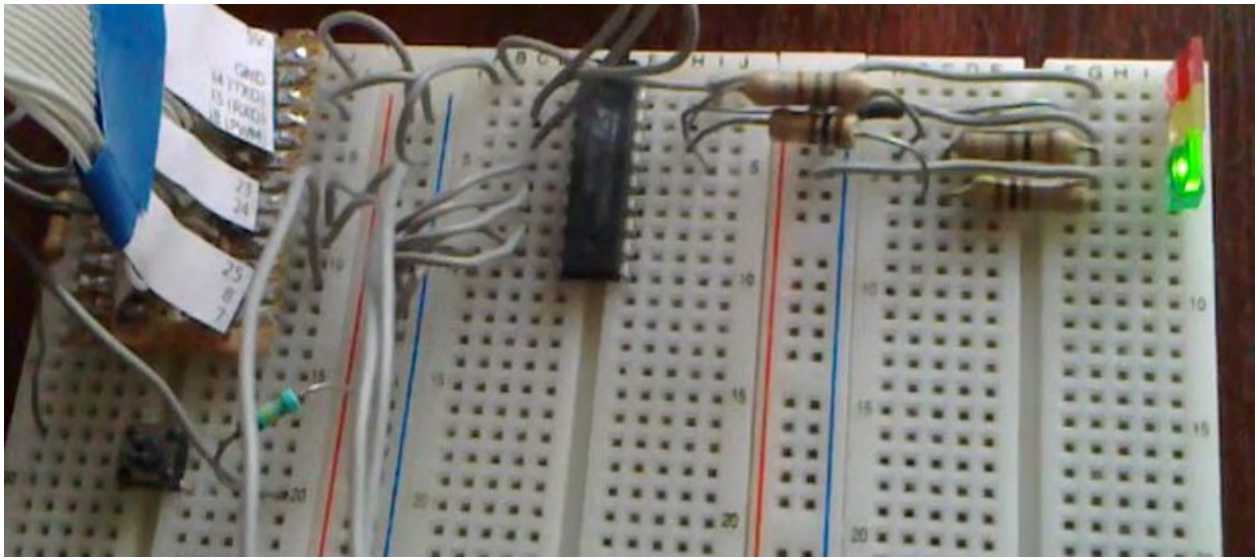
wiringPi Installation (>>)

```
pi@raspbmc:~$ sudo apt-get install git-core make gcc
pi@raspbmc:~$ git clone git://git.drogon.net/wiringPi
pi@raspbmc:~$ cd wiringPi
pi@raspbmc:~/wiringPi$ git pull origin
pi@raspbmc:~/wiringPi$ ./build

pi@raspbmc:~/wiringPi$ sudo su
# echo "/usr/local/lib" >/etc/ld.so.conf.d/local.conf
```

Beispiel: Ampelsteuerung

Beispiel: Ampelsteuerung



Beispiel: Ampelsteuerung

- als Shell-Script (trafficlight.sh)
- 1 GPIO Pin als Input/Interrupt
- 3 GPIO Pins als Outputs

Verwendete Hardware

- Taster
- Widerstände (1x 10k, 1x 100k)
- Treiber-IC: 74HC541
- LEDs + Vorwiderstand (je 3x)

Verwendete Hardware

- Taster
- Widerstände (1x 10k, 1x 100k)
- Treiber-IC: 74HC541
- LEDs + Vorwiderstand (je 3x)

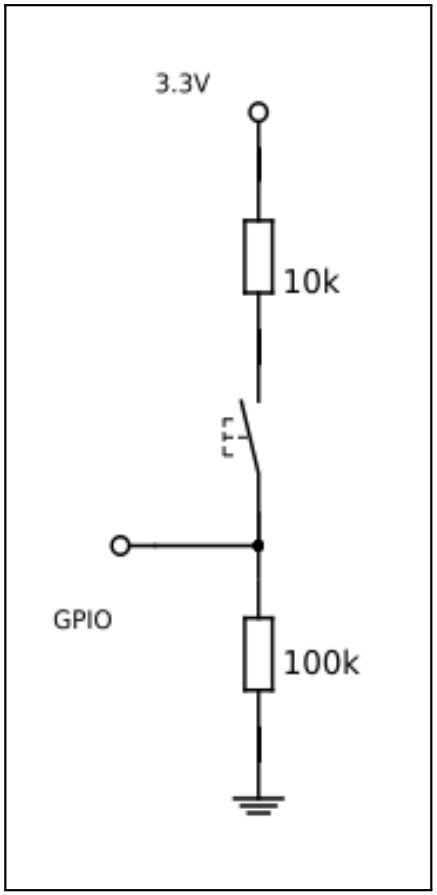
Verwendete Hardware

- Taster
- Widerstände (1x 10k, 1x 100k)
- Treiber-IC: 74HC541
- LEDs + Vorwiderstand (je 3x)

Verwendete Hardware

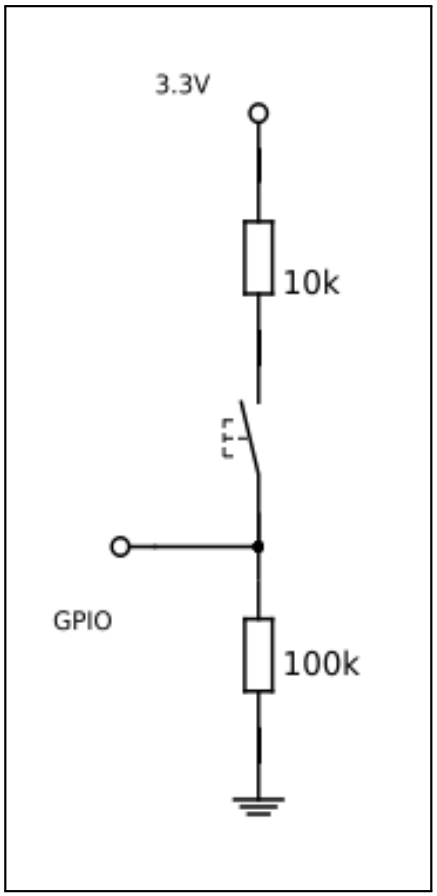
- Taster
- Widerstände (1x 10k, 1x 100k)
- Treiber-IC: 74HC541
- LEDs + Vorwiderstand (je 3x)

Taster: Verdrahtung



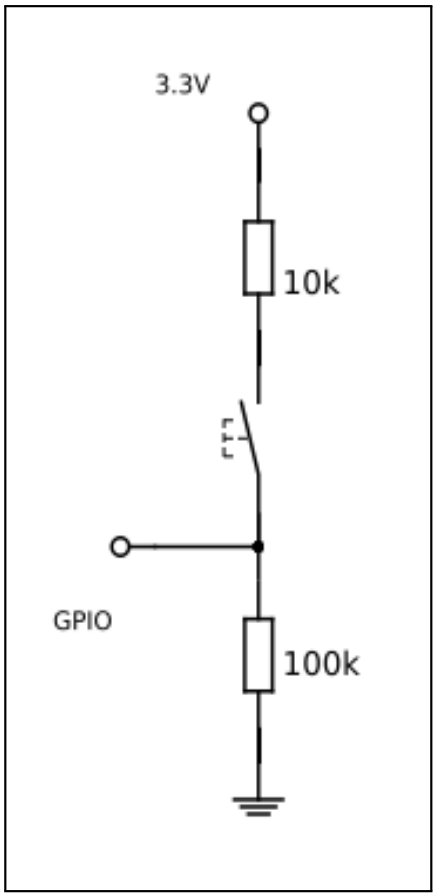
- 100k „pull down“, damit Input nicht „offen“ ist.
- 10k „pull up“, falls Pin irrtümlich als Output konfiguriert ist.
- ... oder man den Pull up irrtümlich an 5V statt 3,3V hängt. :-)

Taster: Verdrahtung



- 100k „pull down“, damit Input nicht „offen“ ist.
- 10k „pull up“, falls Pin irrtümlich als Output konfiguriert ist.
- ... oder man den Pull up irrtümlich an 5V statt 3,3V hängt. :-)

Taster: Verdrahtung



- 100k „pull down“, damit Input nicht „offen“ ist.
- 10k „pull up“, falls Pin irrtümlich als Output konfiguriert ist.
- ... oder man den Pull up irrtümlich an 5V statt 3,3V hängt. :-)

Ausgang: Warum der 74HC541 Treiber?

- Entlastung der GPIO Pins
- Ausgangsspannung (3,3V) kommt von einem integrierten Mini-Netzteil
- max. Ausgangsstrom per GPIO Pin: 16mA
- max. Ausgangsstrom gesamt: 50mA
- Daher 74HC541 mit 5V vom „großen“ Netzteil betreiben
- Die 0V / 3,3V an den Eingängen werden als logische 0 / 1 erkannt

Ausgang: Warum der 74HC541 Treiber?

- Entlastung der GPIO Pins
- Ausgangsspannung (3,3V) kommt von einem integrierten Mini-Netzteil
- max. Ausgangsstrom per GPIO Pin: 16mA
- max. Ausgangsstrom gesamt: 50mA
- Daher 74HC541 mit 5V vom „großen“ Netzteil betreiben
- Die 0V / 3,3V an den Eingängen werden als logische 0 / 1 erkannt

Ausgang: Warum der 74HC541 Treiber?

- Entlastung der GPIO Pins
- Ausgangsspannung (3,3V) kommt von einem integrierten Mini-Netzteil
- max. Ausgangsstrom per GPIO Pin: 16mA
- max. Ausgangsstrom gesamt: 50mA
- Daher 74HC541 mit 5V vom „großen“ Netzteil betreiben
- Die 0V / 3,3V an den Eingängen werden als logische 0 / 1 erkannt

Ausgang: Warum der 74HC541 Treiber?

- Entlastung der GPIO Pins
- Ausgangsspannung (3,3V) kommt von einem integrierten Mini-Netzteil
- max. Ausgangsstrom per GPIO Pin: 16mA
- max. Ausgangsstrom gesamt: 50mA
- Daher 74HC541 mit 5V vom „großen“ Netzteil betreiben
- Die 0V / 3,3V an den Eingängen werden als logische 0 / 1 erkannt

Ausgang: Warum der 74HC541 Treiber?

- Entlastung der GPIO Pins
- Ausgangsspannung (3,3V) kommt von einem integrierten Mini-Netzteil
- max. Ausgangsstrom per GPIO Pin: 16mA
- max. Ausgangsstrom gesamt: 50mA
- Daher 74HC541 mit 5V vom „großen“ Netzteil betreiben
- Die 0V / 3,3V an den Eingängen werden als logische 0 / 1 erkannt

Ausgang: Warum der 74HC541 Treiber?

- Entlastung der GPIO Pins
- Ausgangsspannung (3,3V) kommt von einem integrierten Mini-Netzteil
- max. Ausgangsstrom per GPIO Pin: 16mA
- max. Ausgangsstrom gesamt: 50mA
- Daher 74HC541 mit 5V vom „großen“ Netzteil betreiben
- Die 0V / 3,3V an den Eingängen werden als logische 0 / 1 erkannt

LEDs (>>)

- Light Emitting Diode
- Anode, Kathode
- einfärbig (IR, rot, gelb, grün, blau)
- mehrfärbig, mit gemeinsamer Anode oder Kathode
- Vorwiderstand zur Strombegrenzung!

Rechenbeispiel: Vorwiderstand für LED (>>)

- Spannung: 5V
- rote LED: 1,6V Spannungsabfall
- (grün: 2,2V, blau: 2,4V)
- Annahme: LED Strom soll 10mA sein
- Ohmsches Gesetz: R = U / I
- rot: R = (5 - 1,6) / 0,01 = 340 Ohm

Script: Auszüge

```
PIN_RED=17
PIN_YELLOW=27
PIN_GREEN=22
PIN_BUTTON=4

GPIO="gpio -g"

$GPIO mode $PIN_RED out
$GPIO mode $PIN_YELLOW out
$GPIO mode $PIN_GREEN out

$GPIO mode $PIN_BUTTON in
```

Script: Auszüge

```
function green
{
    $GPIO write $PIN_RED 0
    $GPIO write $PIN_YELLOW 0
    $GPIO write $PIN_GREEN 1

    $GPIO wfi $PIN_BUTTON rising
}
```

Script: Auszüge

```
while true
do
    green
    green_blink
    yellow
    red
    red_yellow
done
```

(Cross-)Compilieren von C-Code

(Cross-)Compilieren von C-Code (>>)

- C, Compiler, Linker, Binary
- nativ: direkt am Pi
- cross: am PC entwickeln und compilieren
- Binary auf den Pi kopieren und ausführen
- `https://github.com/raspberrypi/tools`
- .../arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/

(Cross-)Compilieren von C-Code (>>)

- C, Compiler, Linker, Binary
- nativ: direkt am Pi
- cross: am PC entwickeln und compilieren
- Binary auf den Pi kopieren und ausführen
- `https://github.com/raspberrypi/tools`
- .../arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/

(Cross-)Compilieren von C-Code (>>)

- C, Compiler, Linker, Binary
- nativ: direkt am Pi
- cross: am PC entwickeln und compilieren
- Binary auf den Pi kopieren und ausführen
- `https://github.com/raspberrypi/tools`
- `.../arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/`

(Cross-)Compilieren von C-Code (>>)

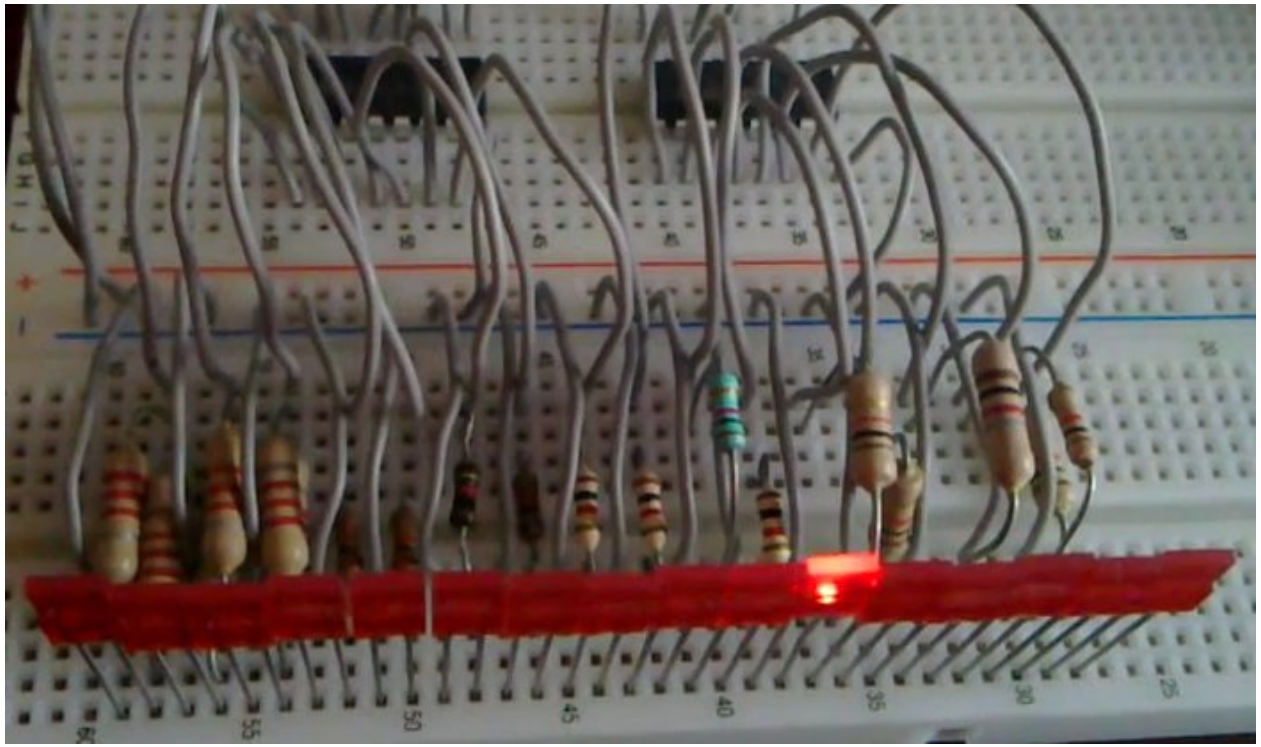
- C, Compiler, Linker, Binary
- nativ: direkt am Pi
- cross: am PC entwickeln und compilieren
- Binary auf den Pi kopieren und ausführen
- `https://github.com/raspberrypi/tools`
- .../arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/

Entwicklungsumgebungen (>>)

- Eclipse: `http://www.gurucoding.com/en/raspberry_pi_eclipse/raspberry_pi_cross_compilation_in_eclipse.php`
- Andere: bitte selber googeln ;-)

Beispiel: KITT meets BSG

Beispiel: KITT meets BSG



Beispiel: KITT meets BSG

- als C Programm
- 3 GPIO Pins als Output
- 2x 74HC595
- 16x LED + Vorwiderstand (1k)

Schieberegister

- 16 LEDs über drei Leitungen ansteuern? Ja, das geht. :-)
- Schieberegister / SIPO (serial in, parallel out)
- 1 Pin für (seriellen) Input
- 1 Pin für Takt (Clock)
- n Pins für (parallelen) Output

Schieberegister

- 16 LEDs über drei Leitungen ansteuern? Ja, das geht. :-)
- Schieberegister / SIPO (serial in, parallel out)
- 1 Pin für (seriellen) Input
- 1 Pin für Takt (Clock)
- n Pins für (parallelen) Output

Schieberegister

- 16 LEDs über drei Leitungen ansteuern? Ja, das geht. :-)
- Schieberegister / SIPO (serial in, parallel out)
- 1 Pin für (seriellen) Input
- 1 Pin für Takt (Clock)
- n Pins für (parallelen) Output

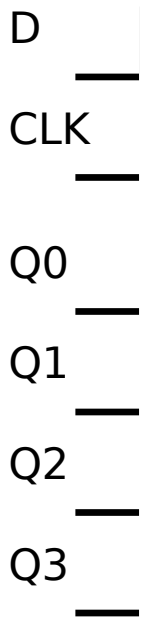
Schieberegister

- 16 LEDs über drei Leitungen ansteuern? Ja, das geht. :-)
- Schieberegister / SIPO (serial in, parallel out)
- 1 Pin für (seriellen) Input
- 1 Pin für Takt (Clock)
- n Pins für (parallelen) Output

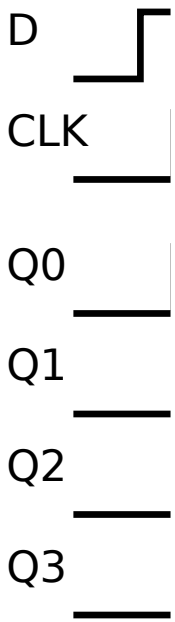
Schieberegister

- 16 LEDs über drei Leitungen ansteuern? Ja, das geht. :-)
- Schieberegister / SIPO (serial in, parallel out)
- 1 Pin für (seriellen) Input
- 1 Pin für Takt (Clock)
- n Pins für (parallelen) Output

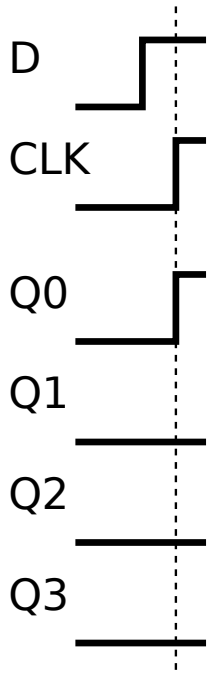
Schieberegister: Timingdiagramm



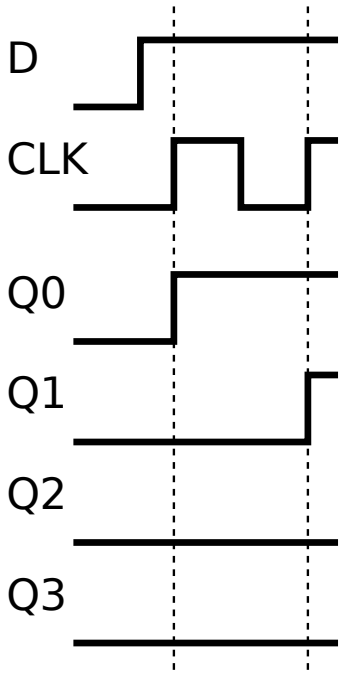
Schieberegister: Timingdiagramm



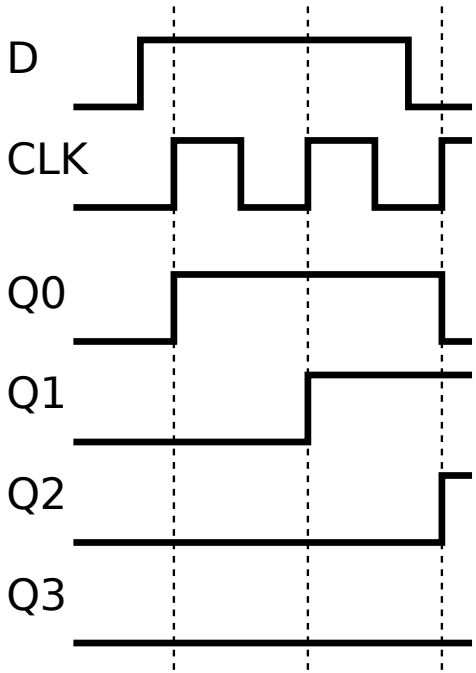
Schieberegister: Timingdiagramm



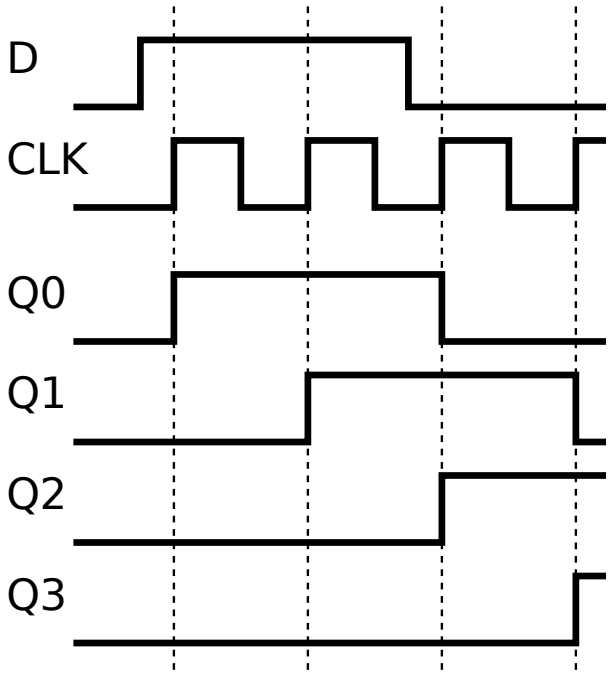
Schieberegister: Timingdiagramm



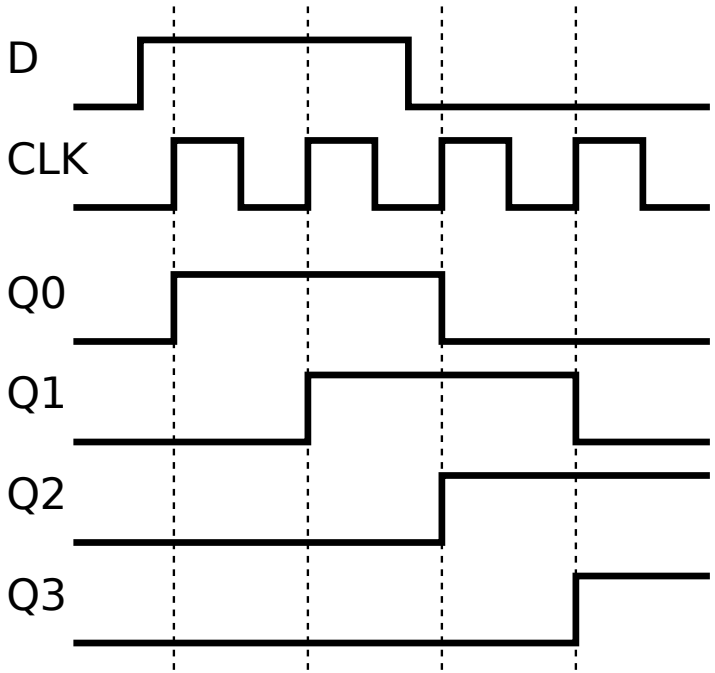
Schieberegister: Timingdiagramm



Schieberegister: Timingdiagramm



Schieberegister: Timingdiagramm



Schieberegister + Latch

- Problem: am Ausgang sind Zwischenzustände sichtbar
- Lösung: 74HC595 ... der Chip aus dem Beispiel
- SI, SCK: Serieller Input, Clock
- QA-QH: 8 Ausgänge
- RCK: Latch Clock (zum „Durchschalten“ der Ausgänge)
- QH': Ausgang zum Kaskadieren
- (verbinden mit SI vom nächsten Chip)

Schieberegister + Latch

- Problem: am Ausgang sind Zwischenzustände sichtbar
- Lösung: 74HC595 ... der Chip aus dem Beispiel
- SI, SCK: Serieller Input, Clock
- QA-QH: 8 Ausgänge
- RCK: Latch Clock (zum „Durchschalten“ der Ausgänge)
- QH': Ausgang zum Kaskadieren
- (verbinden mit SI vom nächsten Chip)

Schieberegister + Latch

- Problem: am Ausgang sind Zwischenzustände sichtbar
- Lösung: 74HC595 ... der Chip aus dem Beispiel
- SI, SCK: Serieller Input, Clock
- QA-QH: 8 Ausgänge
- RCK: Latch Clock (zum „Durchschalten“ der Ausgänge)
- QH': Ausgang zum Kaskadieren
- (verbinden mit SI vom nächsten Chip)

Schieberegister + Latch

- Problem: am Ausgang sind Zwischenzustände sichtbar
- Lösung: 74HC595 ... der Chip aus dem Beispiel
- SI, SCK: Serieller Input, Clock
- QA-QH: 8 Ausgänge
- RCK: Latch Clock (zum „Durchschalten“ der Ausgänge)
- QH': Ausgang zum Kaskadieren
- (verbinden mit SI vom nächsten Chip)

Schieberegister + Latch

- Problem: am Ausgang sind Zwischenzustände sichtbar
- Lösung: 74HC595 ... der Chip aus dem Beispiel
- SI, SCK: Serieller Input, Clock
- QA-QH: 8 Ausgänge
- RCK: Latch Clock (zum „Durchschalten“ der Ausgänge)
- QH': Ausgang zum Kaskadieren
- (verbinden mit SI vom nächsten Chip)

Schieberegister + Latch

- Problem: am Ausgang sind Zwischenzustände sichtbar
- Lösung: 74HC595 ... der Chip aus dem Beispiel
- SI, SCK: Serieller Input, Clock
- QA-QH: 8 Ausgänge
- RCK: Latch Clock (zum „Durchschalten“ der Ausgänge)
- QH': Ausgang zum Kaskadieren
- (verbinden mit SI vom nächsten Chip)

Schieberegister + Latch

- Problem: am Ausgang sind Zwischenzustände sichtbar
- Lösung: 74HC595 ... der Chip aus dem Beispiel
- SI, SCK: Serieller Input, Clock
- QA-QH: 8 Ausgänge
- RCK: Latch Clock (zum „Durchschalten“ der Ausgänge)
- QH': Ausgang zum Kaskadieren
- (verbinden mit SI vom nächsten Chip)

wiringPi / sr595.h

- Initialisierung von drei beliebigen Pins (Data, Clock, Latch)
- Definition der Anzahl der Ausgangspins
- Abbildung auf „virtuelle“ Ausgangspins
- Komplette Abstraktion der Kommunikation mit dem Chip

wiringPi / sr595.h

- Initialisierung von drei beliebigen Pins (Data, Clock, Latch)
- Definition der Anzahl der Ausgangspins
- Abbildung auf „virtuelle“ Ausgangspins
- Komplette Abstraktion der Kommunikation mit dem Chip

wiringPi / sr595.h

- Initialisierung von drei beliebigen Pins (Data, Clock, Latch)
- Definition der Anzahl der Ausgangspins
- Abbildung auf „virtuelle“ Ausgangspins
- Komplette Abstraktion der Kommunikation mit dem Chip

wiringPi / sr595.h

- Initialisierung von drei beliebigen Pins (Data, Clock, Latch)
- Definition der Anzahl der Ausgangspins
- Abbildung auf „virtuelle“ Ausgangspins
- Komplette Abstraktion der Kommunikation mit dem Chip

C Source: Auszüge

```
#include "wiringPi.h"
#include "sr595.h"

#define PIN_BASE 100
#define PIN_DATA 23
#define PIN_CLOCK 24
#define PIN_LATCH 25

#define LEDS 16
```

C Source: Auszüge

```
int rc;

rc = wiringPiSetupGpio();
if (rc != 0) {
    // fail
}

rc = sr595Setup(PIN_BASE, LEDS,
    PIN_DATA, PIN_CLOCK, PIN_LATCH);
if (rc != 0) {
    // fail
}
```

C Source: Auszüge

```
int activeBit = 0;
int increment = 1;

for (;;) {
    for (int led=0; led<LEDS; led++) {
        digitalWrite(PIN_BASE + led, activeBit == led ? 1 : 0);
    }
    activeBit += increment;
    if (activeBit == 0 || activeBit == LEDS-1) {
        increment *= -1;
    }
    delay(80);
}
```

74HC595 Nachteile

- max. Strom (35mA pro Pin, 70mA gesamt)
- 74HCxxx sind Logik-ICs und nicht für große Lasten gedacht
- High-power Variante: z.B. TPIC6B595
- Oder für LEDs: spezielle Treiber-Chips

74HC595 Nachteile

- max. Strom (35mA pro Pin, 70mA gesamt)
- 74HCxxx sind Logik-ICs und nicht für große Lasten gedacht
- High-power Variante: z.B. TPIC6B595
- Oder für LEDs: spezielle Treiber-Chips

74HC595 Nachteile

- max. Strom (35mA pro Pin, 70mA gesamt)
- 74HCxxx sind Logik-ICs und nicht für große Lasten gedacht
- High-power Variante: z.B. TPIC6B595
- Oder für LEDs: spezielle Treiber-Chips

74HC595 Nachteile

- max. Strom (35mA pro Pin, 70mA gesamt)
- 74HCxxx sind Logik-ICs und nicht für große Lasten gedacht
- High-power Variante: z.B. TPIC6B595
- Oder für LEDs: spezielle Treiber-Chips

LED Driver (constant current sink)

- Vorteil: Nur ein Widerstand pro Chip
- ... mit dem der Strom für alle Ausgänge eingestellt wird
- LEDs können ohne Vorwiderstand angeschlossen werden
- MBI5026
- STP08DP05 / CP05
- TLC5940, 5916, 5925
- Problem: die Auswahl an DIP-Chips ist sehr begrenzt :-)
- Problem: kaum bis gar nicht im Elektronik-Laden erhältlich

LED Driver (constant current sink)

- Vorteil: Nur ein Widerstand pro Chip
- ... mit dem der Strom für alle Ausgänge eingestellt wird
- LEDs können ohne Vorwiderstand angeschlossen werden
- MBI5026
- STP08DP05 / CP05
- TLC5940, 5916, 5925
- Problem: die Auswahl an DIP-Chips ist sehr begrenzt :-)
- Problem: kaum bis gar nicht im Elektronik-Laden erhältlich

LED Driver (constant current sink)

- Vorteil: Nur ein Widerstand pro Chip
- ... mit dem der Strom für alle Ausgänge eingestellt wird
- LEDs können ohne Vorwiderstand angeschlossen werden
- MBI5026
- STP08DP05 / CP05
- TLC5940, 5916, 5925
- Problem: die Auswahl an DIP-Chips ist sehr begrenzt :-)
- Problem: kaum bis gar nicht im Elektronik-Laden erhältlich

LED Driver (constant current sink)

- Vorteil: Nur ein Widerstand pro Chip
- ... mit dem der Strom für alle Ausgänge eingestellt wird
- LEDs können ohne Vorwiderstand angeschlossen werden
- MBI5026
- STP08DP05 / CP05
- TLC5940, 5916, 5925
- Problem: die Auswahl an DIP-Chips ist sehr begrenzt :-)
- Problem: kaum bis gar nicht im Elektronik-Laden erhältlich

LED Driver (constant current sink)

- Vorteil: Nur ein Widerstand pro Chip
- ... mit dem der Strom für alle Ausgänge eingestellt wird
- LEDs können ohne Vorwiderstand angeschlossen werden
- MBI5026
- STP08DP05 / CP05
- TLC5940, 5916, 5925
- Problem: die Auswahl an DIP-Chips ist sehr begrenzt :-)
- Problem: kaum bis gar nicht im Elektronik-Laden erhältlich

LED Driver (constant current sink)

- Vorteil: Nur ein Widerstand pro Chip
- ... mit dem der Strom für alle Ausgänge eingestellt wird
- LEDs können ohne Vorwiderstand angeschlossen werden
- MBI5026
- STP08DP05 / CP05
- TLC5940, 5916, 5925
- Problem: die Auswahl an DIP-Chips ist sehr begrenzt :-)
- Problem: kaum bis gar nicht im Elektronik-Laden erhältlich

LED Driver (constant current sink)

- Vorteil: Nur ein Widerstand pro Chip
- ... mit dem der Strom für alle Ausgänge eingestellt wird
- LEDs können ohne Vorwiderstand angeschlossen werden
- MBI5026
- STP08DP05 / CP05
- TLC5940, 5916, 5925
- Problem: die Auswahl an DIP-Chips ist sehr begrenzt :-)
- Problem: kaum bis gar nicht im Elektronik-Laden erhältlich

LED Driver (constant current sink)

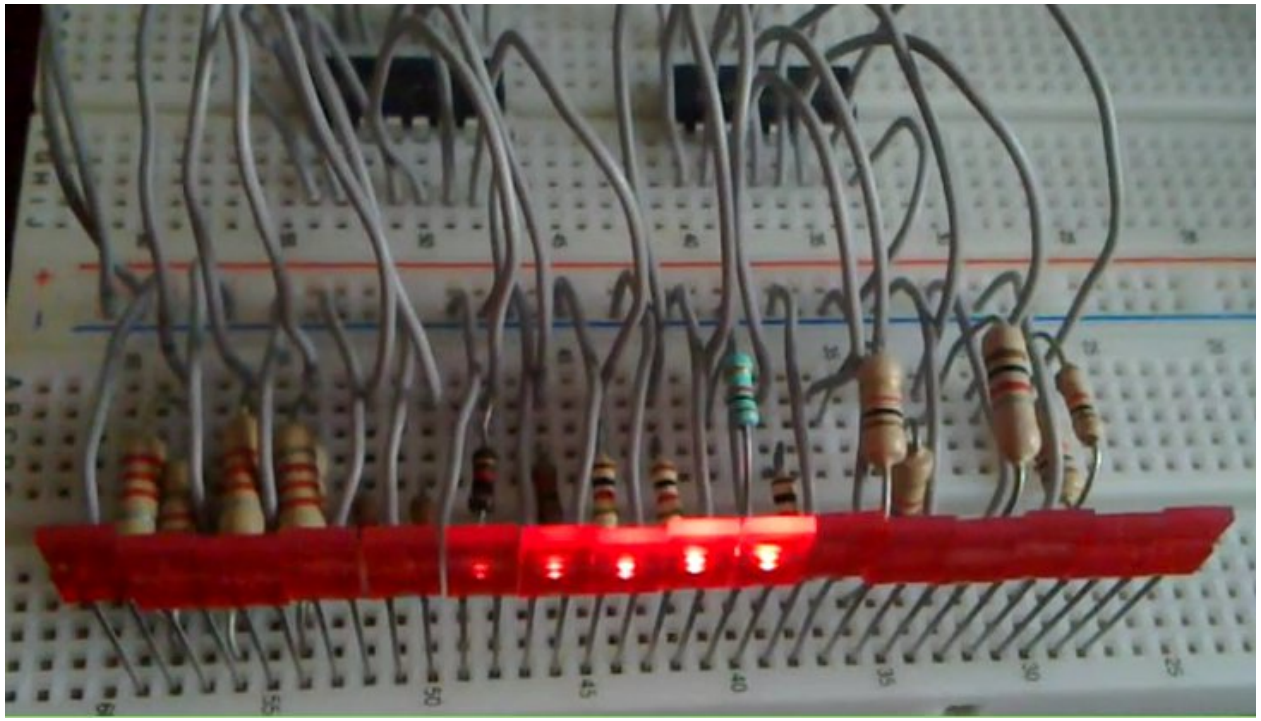
- Vorteil: Nur ein Widerstand pro Chip
- ... mit dem der Strom für alle Ausgänge eingestellt wird
- LEDs können ohne Vorwiderstand angeschlossen werden
- MBI5026
- STP08DP05 / CP05
- TLC5940, 5916, 5925
- Problem: die Auswahl an DIP-Chips ist sehr begrenzt :-)
- Problem: kaum bis gar nicht im Elektronik-Laden erhältlich

LED Driver: Demo-Videos

- 1x TLC5940, 16 rote LEDs
- 2x TLC5940, 10 RGB LEDs

Beispiel: Helligkeitssteuerung

Beispiel: Pulsbreitenmodulationszylone



PWM als Helligkeitssteuerung

- Digitale Ausgänge können nur 1 oder 0 sein
- PWM: Pulse Width Modulation
- Kontinuierliches Umschalten zwischen 1 und 0
- Das Tastverhältnis zwischen 1 und 0 ergibt eine „Helligkeitsstufe“

PWM als Helligkeitssteuerung

- Digitale Ausgänge können nur 1 oder 0 sein
- PWM: Pulse Width Modulation
- Kontinuierliches Umschalten zwischen 1 und 0
- Das Tastverhältnis zwischen 1 und 0 ergibt eine „Helligkeitsstufe“

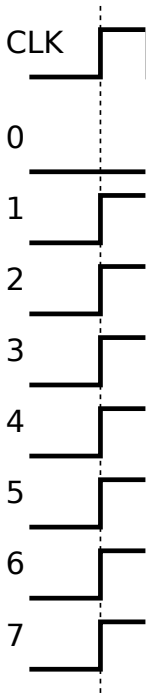
PWM als Helligkeitssteuerung

- Digitale Ausgänge können nur 1 oder 0 sein
- PWM: Pulse Width Modulation
- Kontinuierliches Umschalten zwischen 1 und 0
- Das Tastverhältnis zwischen 1 und 0 ergibt eine „Helligkeitsstufe“

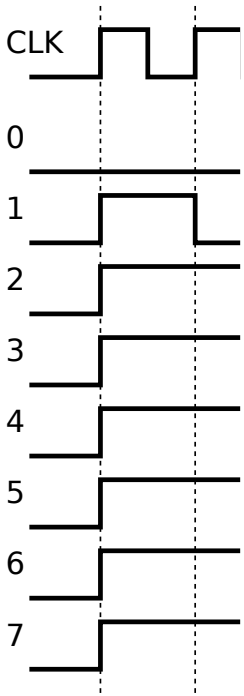
PWM als Helligkeitssteuerung

- Digitale Ausgänge können nur 1 oder 0 sein
- PWM: Pulse Width Modulation
- Kontinuierliches Umschalten zwischen 1 und 0
- Das Tastverhältnis zwischen 1 und 0 ergibt eine „Helligkeitsstufe“

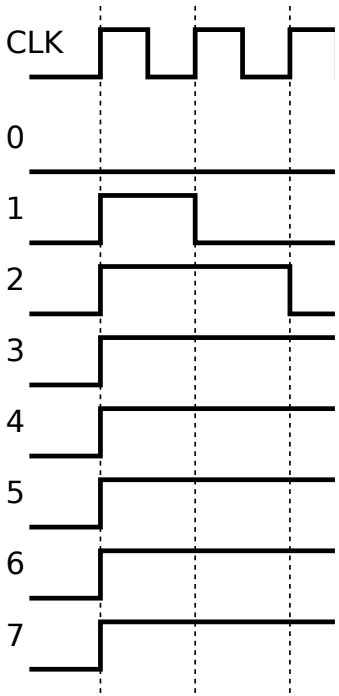
PWM: Timingdiagramm



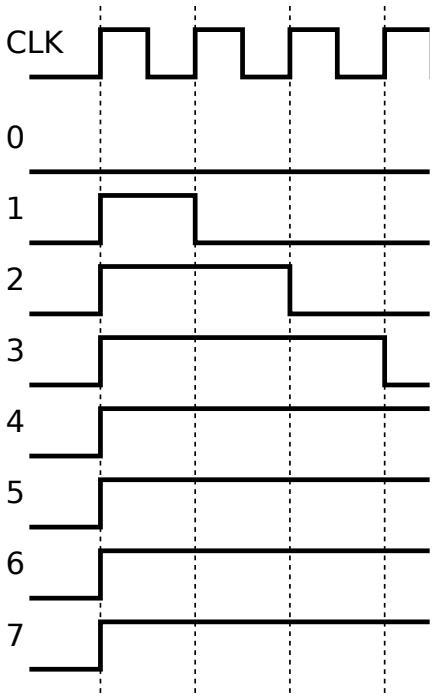
PWM: Timingdiagramm



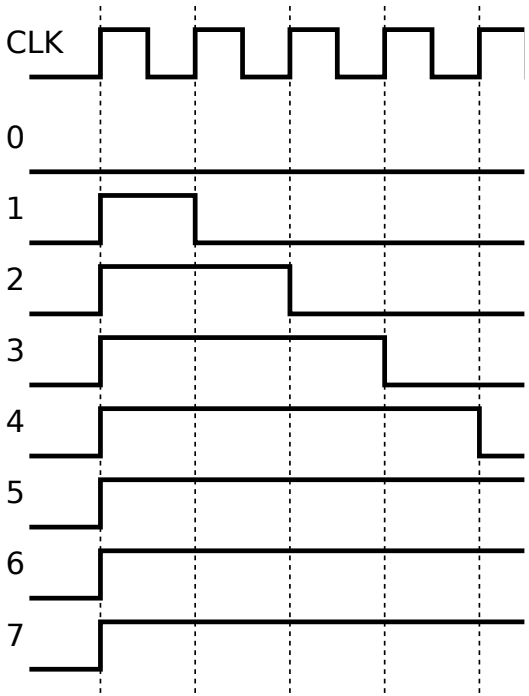
PWM: Timingdiagramm



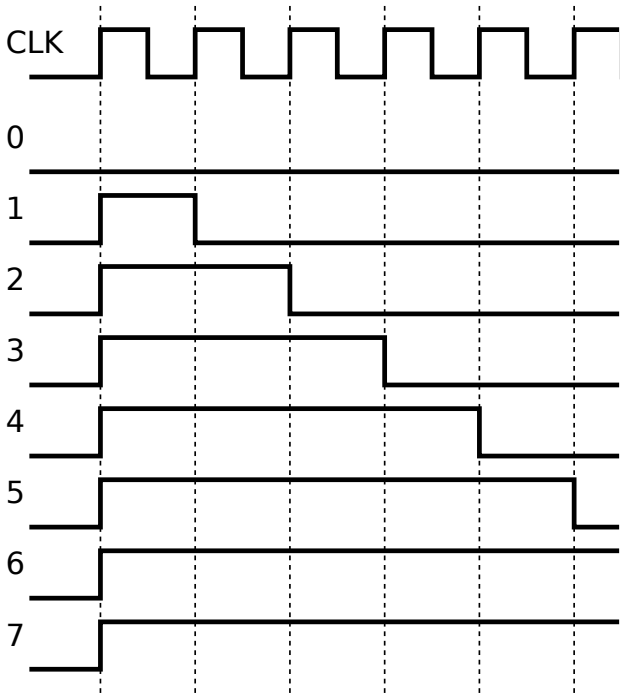
PWM: Timingdiagramm



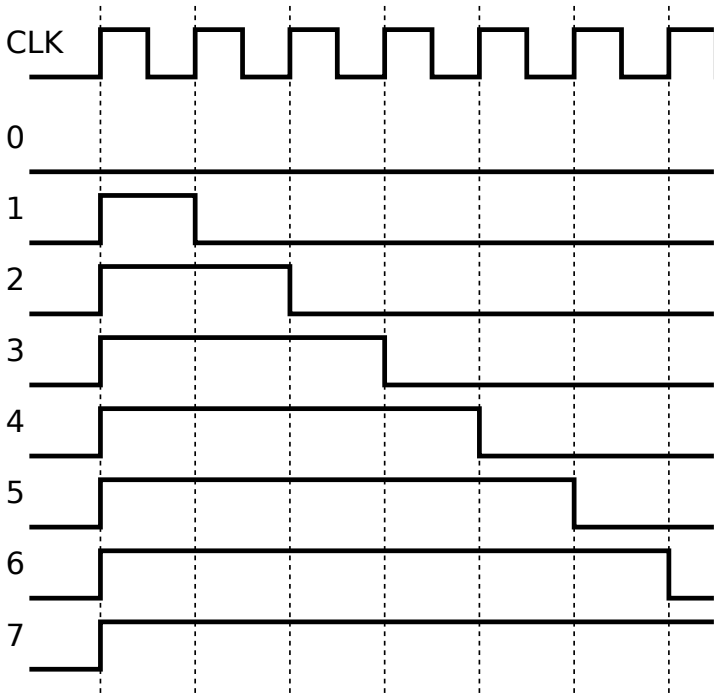
PWM: Timingdiagramm



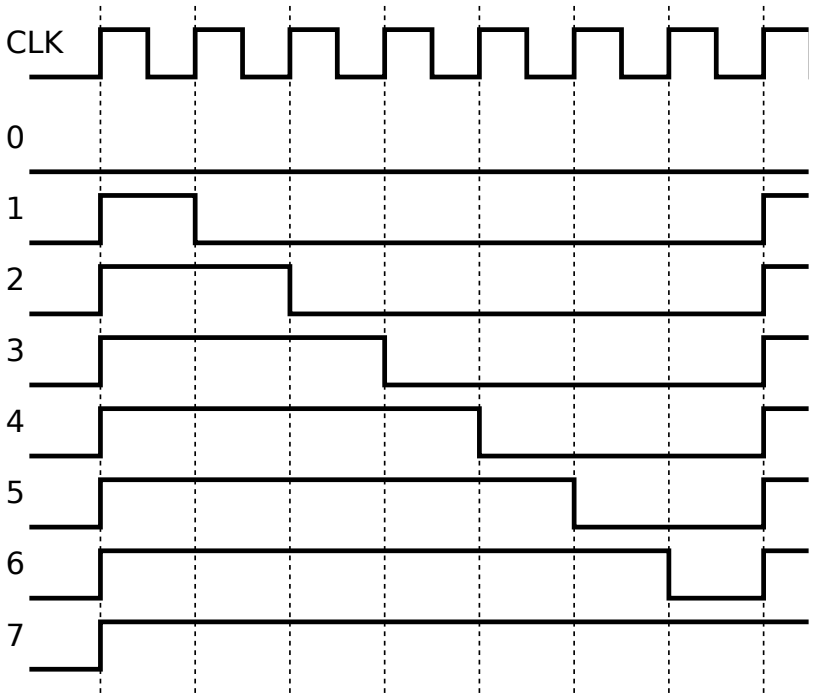
PWM: Timingdiagramm



PWM: Timingdiagramm



PWM: Timingdiagramm



Software-PWM

- Raspberry Pi hat nur einen PWM Pin
- PWM kann auch in Software implementiert werden
- Erfordert kontinuierliche digitalWrite()s
- Timing ist wichtig (ARM/Linux vs. Arduino)
- wiringPi: softPwm.h - für einzelne Pins
- nicht brauchbar wenn man hinter einem Schieberegister PWM haben will.

Software-PWM

- Raspberry Pi hat nur einen PWM Pin
- PWM kann auch in Software implementiert werden
- Erfordert kontinuierliche digitalWrite()s
- Timing ist wichtig (ARM/Linux vs. Arduino)
- wiringPi: softPwm.h - für einzelne Pins
- nicht brauchbar wenn man hinter einem Schieberegister PWM haben will.

Software-PWM

- Raspberry Pi hat nur einen PWM Pin
- PWM kann auch in Software implementiert werden
- Erfordert kontinuierliche digitalWrite()s
- Timing ist wichtig (ARM/Linux vs. Arduino)
- wiringPi: softPwm.h - für einzelne Pins
- nicht brauchbar wenn man hinter einem Schieberegister PWM haben will.

Software-PWM

- Raspberry Pi hat nur einen PWM Pin
- PWM kann auch in Software implementiert werden
- Erfordert kontinuierliche digitalWrite()s
- Timing ist wichtig (ARM/Linux vs. Arduino)
- wiringPi: softPwm.h - für einzelne Pins
- nicht brauchbar wenn man hinter einem Schieberegister PWM haben will.

Software-PWM

- Raspberry Pi hat nur einen PWM Pin
- PWM kann auch in Software implementiert werden
- Erfordert kontinuierliche digitalWrite()s
- Timing ist wichtig (ARM/Linux vs. Arduino)
- wiringPi: softPwm.h - für einzelne Pins
- nicht brauchbar wenn man hinter einem Schieberegister PWM haben will.

Software-PWM

- Raspberry Pi hat nur einen PWM Pin
- PWM kann auch in Software implementiert werden
- Erfordert kontinuierliche digitalWrite()s
- Timing ist wichtig (ARM/Linux vs. Arduino)
- wiringPi: softPwm.h - für einzelne Pins
- nicht brauchbar wenn man hinter einem Schieberegister PWM haben will.

C Source: Auszüge

```
#define FRAMES 60
#define LEDS 16

int frames[FRAMES][LEDS] = {
    ...
    { 1, 2, 4, 6, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    ...
    { 0, 0, 1, 2, 4, 6, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    ...
    { 0, 0, 0, 0, 1, 2, 4, 6, 7, 0, 0, 0, 0, 0, 0, 0 },
    ...
};
```

C Source: Auszüge

```
for (int frame = 0; frame < FRAMES; frame++) {
    for (int loop = 0; loop < PWM_LOOPS; loop++) {
        for (int level = 0; level < 7; level++) {
            for (int led = 0; led < LEDS; led++) {
                digitalWrite(PIN_BASE + led,
                    frames[frame][led] > level ? 1 : 0);
            }
        }
    }
}
```

wiringPi / Performance von sr595.h

- Bei 16 LEDs / 8 Helligkeitsstufen: 110 „fps“
- sr595-Code enthält delay()s beim Hinaustakten
- Laufzeit steigt quadratisch mit der Anzahl der Ausgangspins
- 595er-Kommunikation selbst geschrieben: 24300 „fps“

wiringPi / Performance von sr595.h

- Bei 16 LEDs / 8 Helligkeitsstufen: 110 „fps“
- sr595-Code enthält delay()s beim Hinaustakten
- Laufzeit steigt quadratisch mit der Anzahl der Ausgangspins
- 595er-Kommunikation selbst geschrieben: 24300 „fps“

wiringPi / Performance von sr595.h

- Bei 16 LEDs / 8 Helligkeitsstufen: 110 „fps“
- sr595-Code enthält delay()s beim Hinaustakten
- Laufzeit steigt quadratisch mit der Anzahl der Ausgangspins
- 595er-Kommunikation selbst geschrieben: 24300 „fps“

wiringPi / Performance von sr595.h

- Bei 16 LEDs / 8 Helligkeitsstufen: 110 „fps“
- sr595-Code enthält delay()s beim Hinaustakten
- Laufzeit steigt quadratisch mit der Anzahl der Ausgangspins
- 595er-Kommunikation selbst geschrieben: 24300 „fps“

C Source: Auszüge

```
pinMode(PIN_DATA, OUTPUT);
pinMode(PIN_CLOCK, OUTPUT);
pinMode(PIN_LATCH, OUTPUT);

digitalWrite(PIN_DATA, 0);
digitalWrite(PIN_CLOCK, 0);
digitalWrite(PIN_LATCH, 0);
```

C Source: Auszüge

```
for (int frame = 0; frame < FRAMES; frame++) {
    for (int loop = 0; loop < PWM_LOOPS; loop++) {
        for (int level = 0; level < 7; level++) {
            for (int led = 0; led < LEDS; led++) {
                digitalWrite(PIN_DATA,
                    frames[frame][led] > level ? 1 : 0);
                digitalWrite(PIN_CLOCK, 1);
                digitalWrite(PIN_CLOCK, 0);
            }
            digitalWrite(PIN_LATCH, 1);
            digitalWrite(PIN_LATCH, 0);
        }
    }
}
```

wiringPi / Grenzen

- Möglich sind ca. 10 Mio. digitalWrite()s / sec.
- oder: 100ns zwischen zwei digitalWrite()s.
- 595: „nur“ 3,3 MBit / sec. (drei digitalWrite()s pro Datenbit)
- Wenn man mehr braucht: improvisieren :-)
- z.B. n 595er parallel (n+2 digitalWrite()s pro n Datenbits)

wiringPi / Grenzen

- Möglich sind ca. 10 Mio. digitalWrite()s / sec.
- oder: 100ns zwischen zwei digitalWrite()s.
- 595: „nur“ 3,3 MBit / sec. (drei digitalWrite()s pro Datenbit)
- Wenn man mehr braucht: improvisieren :-)
- z.B. n 595er parallel (n+2 digitalWrite()s pro n Datenbits)

wiringPi / Grenzen

- Möglich sind ca. 10 Mio. digitalWrite()s / sec.
- oder: 100ns zwischen zwei digitalWrite()s.
- 595: „nur“ 3,3 MBit / sec. (drei digitalWrite()s pro Datenbit)
- Wenn man mehr braucht: improvisieren :-)
- z.B. n 595er parallel (n+2 digitalWrite()s pro n Datenbits)

wiringPi / Grenzen

- Möglich sind ca. 10 Mio. digitalWrite()s / sec.
- oder: 100ns zwischen zwei digitalWrite()s.
- 595: „nur“ 3,3 MBit / sec. (drei digitalWrite()s pro Datenbit)
- Wenn man mehr braucht: improvisieren :-)
- z.B. n 595er parallel (n+2 digitalWrite()s pro n Datenbits)

wiringPi / Grenzen

- Möglich sind ca. 10 Mio. digitalWrite()s / sec.
- oder: 100ns zwischen zwei digitalWrite()s.
- 595: „nur“ 3,3 MBit / sec. (drei digitalWrite()s pro Datenbit)
- Wenn man mehr braucht: improvisieren :-)
- z.B. n 595er parallel (n+2 digitalWrite()s pro n Datenbits)

To be continued...

RGB LED Schneeflocke

- für die kommenden Weihnachten
- 175 RGB LEDs (common anode)
- in einer 40x40cm Plexiglasscheibe
- 6x TLC5940 (je zwei für rot/grün/blau)
- 6x NDP6020P MOSFETs, 1x 74HC138 (für Multiplexing)
- ...

RGB LED Schneeflocke

- für die kommenden Weihnachten
- 175 RGB LEDs (common anode)
- in einer 40x40cm Plexiglasscheibe
- 6x TLC5940 (je zwei für rot/grün/blau)
- 6x NDP6020P MOSFETs, 1x 74HC138 (für Multiplexing)
- ...

RGB LED Schneeflocke

- für die kommenden Weihnachten
- 175 RGB LEDs (common anode)
- in einer 40x40cm Plexiglasscheibe
- 6x TLC5940 (je zwei für rot/grün/blau)
- 6x NDP6020P MOSFETs, 1x 74HC138 (für Multiplexing)
- ...

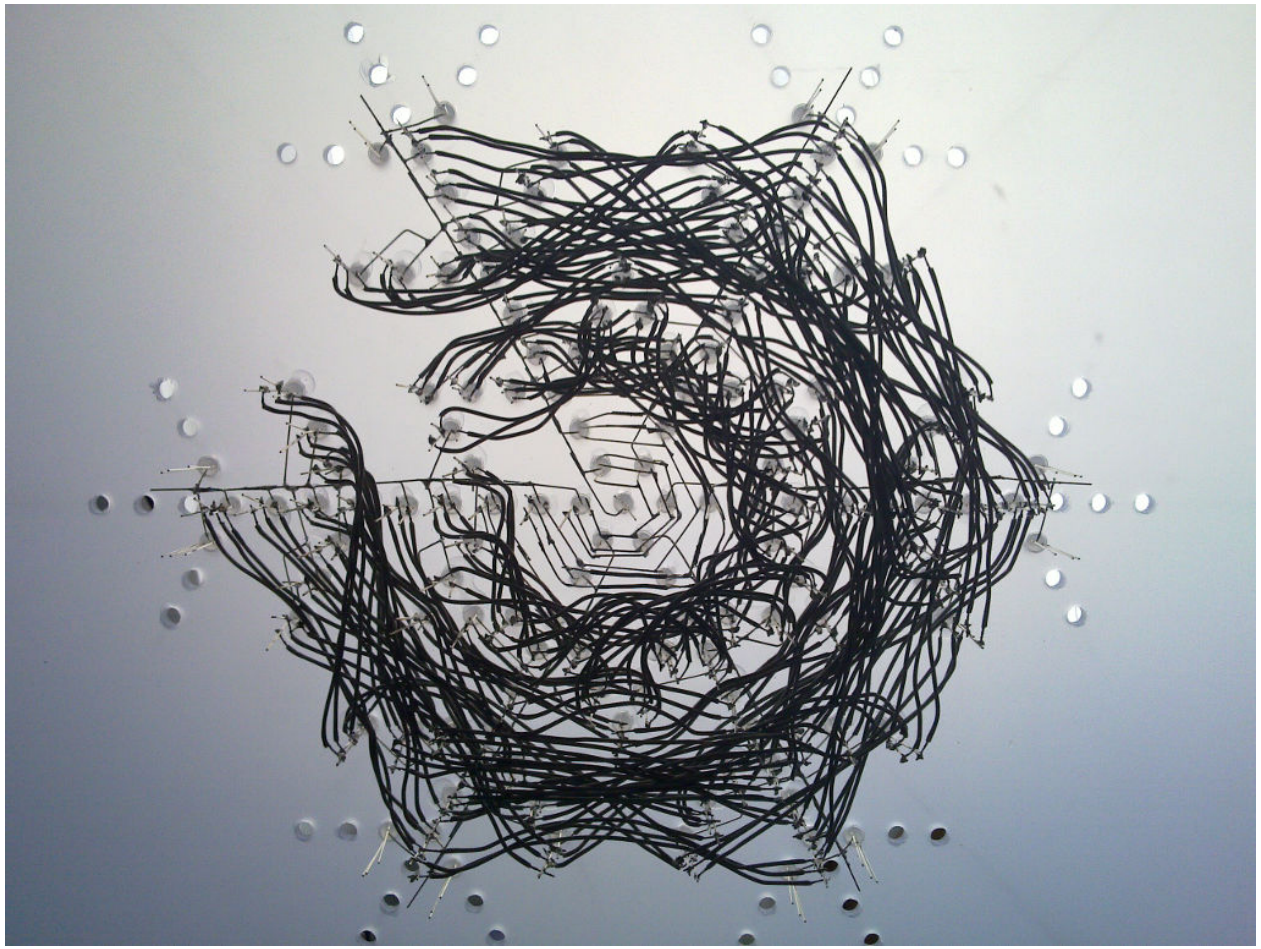
RGB LED Schneeflocke

- für die kommenden Weihnachten
- 175 RGB LEDs (common anode)
- in einer 40x40cm Plexiglasscheibe
- 6x TLC5940 (je zwei für rot/grün/blau)
- 6x NDP6020P MOSFETs, 1x 74HC138 (für Multiplexing)
- ...

RGB LED Schneeflocke

- für die kommenden Weihnachten
- 175 RGB LEDs (common anode)
- in einer 40x40cm Plexiglasscheibe
- 6x TLC5940 (je zwei für rot/grün/blau)
- 6x NDP6020P MOSFETs, 1x 74HC138 (für Multiplexing)
- ...

RGB LED Schneeflocke



Q & A