Einstieg in den hardwarenahen Unterricht mit dem Raspberry Pi

andre.maier@elektronikschule.de

Ziele der Fortbildung

- Überblick und techn. Hintergrundwissen zur GPIO-Schnittstelle des Raspberry Pi (sowohl hardware- als auch softwareseitig)
- Exemplarische Betrachtung der grundsätzlichen Ansteuerung auf verschiedenen Ebenen und in verschiedenen Programmiersprachen
- Beleuchtung der für eine Einführung im Unterricht relevanten fachwissenschaftlichen und didaktischen Zusammenhänge
- Diskussion und Erfahrungsaustausch

Materialien zur Fortbildung

http://www.elektronikschule.de/~maierm/

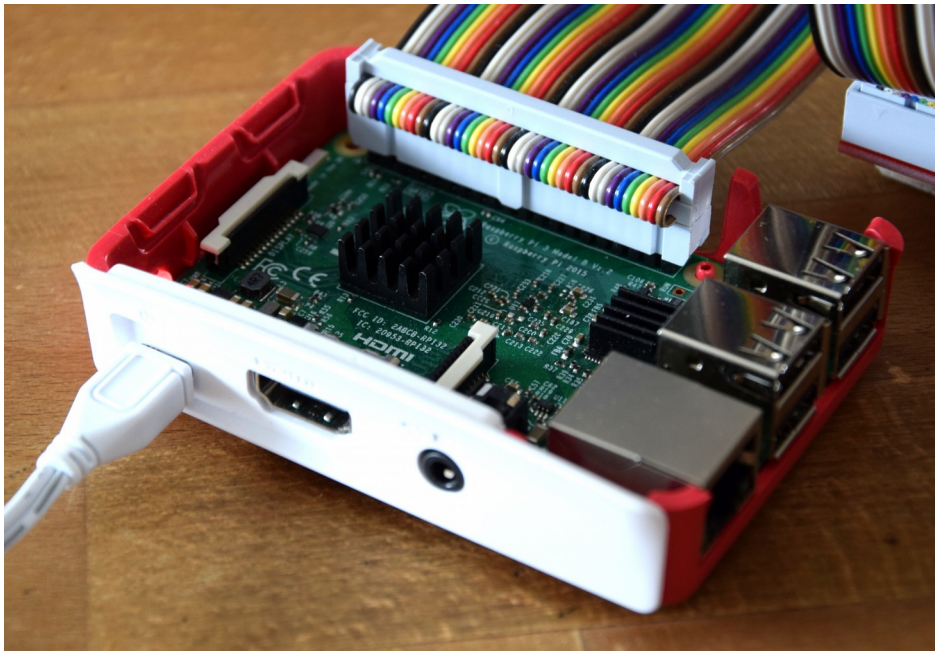
→ Lecture Notes

→ Lehrerfortbildung

→ raspberrypi_gpio_einfuehrung_rpt/

Ein paar Zahlen und Fakten zum Raspberry Pi 3

- Broadcom BCM2837 Chip
  - CPU: 1.2 GHz 64-bit quad-core ARM Cortex A53
  - GPU: Broadcom VideoCore IV
  - RAM: 1 GB LPDDR2 (900 MHz)
- Ext. Speicher für Betriebssystem: microSD
- Netzwerkschnittstellen
  - 2.4 GHz 802.11n wireless
  - Ethernet (100 Mbit/s bzw. 10 Mbit/s)
- Weitere Schnittstellen
  - 4 x USB 2.0
  - HDMI
  - 3.5mm Audio (+ Composite Video) - Ausgang
  - Micro-USB zur Stromversorgung
  - GPIO-Schnittstelle



Vorteile des Raspberry Pi im Unterricht

- Extrem hohe Abdeckung von Lehrplaninhalten mit *einem einzigen* System
- Möglichkeit der Anpassung des Unterrichts auf die Schulart und Jahrgangsstufe in einem weiten Bereich
- Individuelles Lernen durch Offenheit des Systems
- Verlangt von den Lernenden und Lehrenden eine stärkere Auseinandersetzung mit der Problemlöse- und Handlungsfähigkeit, als bloße der Reproduktion von Faktenwissen
- System ist für die Schülerinnen und Schüler erschwinglich
- Zahlreiche Beispielprojekte, Tutorials und Foren im Netz verfügbar
- Geeignet für Projektarbeiten
- System hat Nutzen über die Schulzeit / den Unterricht hinaus

Ungefähre* Kosten für die SuS

Artikel	Preis in Euro
Raspberry Pi 3 Modell B	36
Gehäuse	9
Netzteil 5 V / 2.5 A	12
SD-Karte 16 GB (San-Disk UHS-1)	10
HDMI-Kabel	5
Breadboard	8
Jumperleitungen-Set	3
Summe	**83**

* Am Beispiel einer Berufskollegsklasse im Herbst 2016. Kleinteile wie Dioden, Widerstände, ICs wurden anderweitig bereitgestellt werden. Die SuS mussten (durften) sich selbst ihr Kit zusammenstellen und eigenverantwortlich bei einem Händler ihrer Wahl bestellen. Der tatsächlich bezahlte Preis für das oben aufgeführte Kit lag im Durchschnitt um 6 Euro niedriger als die genannte Abschätzung. Der Beschaffungsprozess wurde von den Schülern dokumentiert. Zuvor wurde per Informationsschreiben durch den Fachlehrer das Einverständnis der Eltern eingeholt.

Einführung des Raspberry Pi im Unterricht

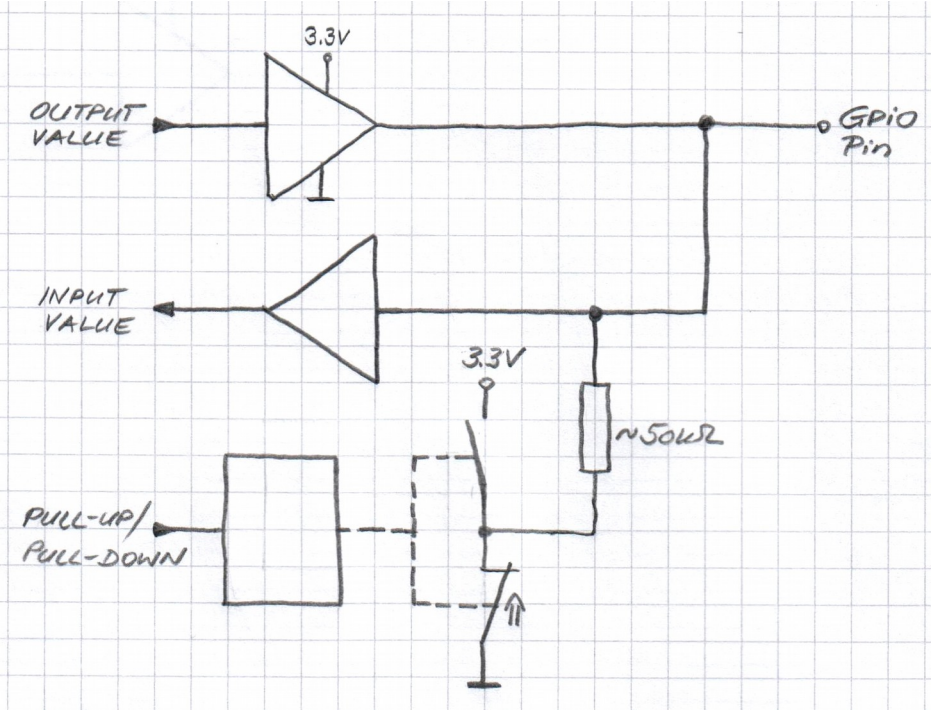
- Ggf. Information der Eltern
- Ggf. Recherche und Beschaffungsprozess
- Installation des Betriebssystems
- Erste Gehversuche mit Linux
  - Betriebssystem, OpenSource, GNU
  - Dateien und Pfade, einfache Kommandos
  - Verwendung eines Texteditors
  - Benutzer und Gruppen, grundlegende Zugriffsrechte

GPIO-Schnittstelle

3.3 V	1	2	5 V
GPIO2	3	4	5 V
GPIO3	5	6	Ground
GPIO4	7	8	GPIO14
Ground	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	Ground
GPIO22	15	16	GPIO23
3.3 V	17	18	GPIO24
GPIO10	19	20	Ground
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
Ground	25	26	GPIO7
GPIO0	27	28	GPIO1
GPIO5	29	30	Ground
GPIO6	31	32	GPIO12
GPIO13	33	34	Ground
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
Ground	39	40	GPIO21

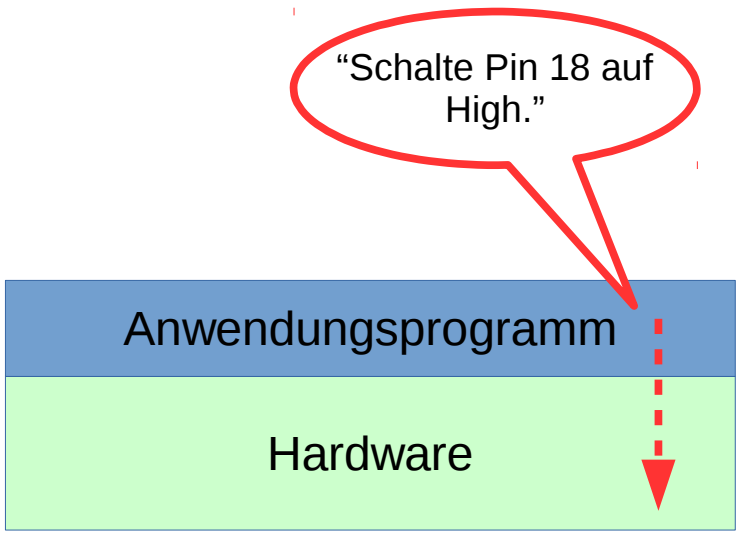
- **GPIO** → **G**eneral **P**urpose **I**nput **O**utput
- bidirektional
- frei programmierbar
- GPIO-Pins arbeiten mit einer Spannung von 3.3 V
- Stromentnahme aus allen 3.3 V - Pins (GPIOs + Pin 1 und 17) soll in der Summe nicht mehr als 50 mA betragen
- Pin 2 und 4 (5 V) sind durch eine selbstrückstellende Sicherung geschützt
- Achtung: Unterschiedliche Angaben je nach Version des BCM283x Bausteins und Informationsquelle. Im Zweifelsfall Datenblatt des Herstellers konsultieren!

Ein (ganz grobes) Ersatzschaltbild

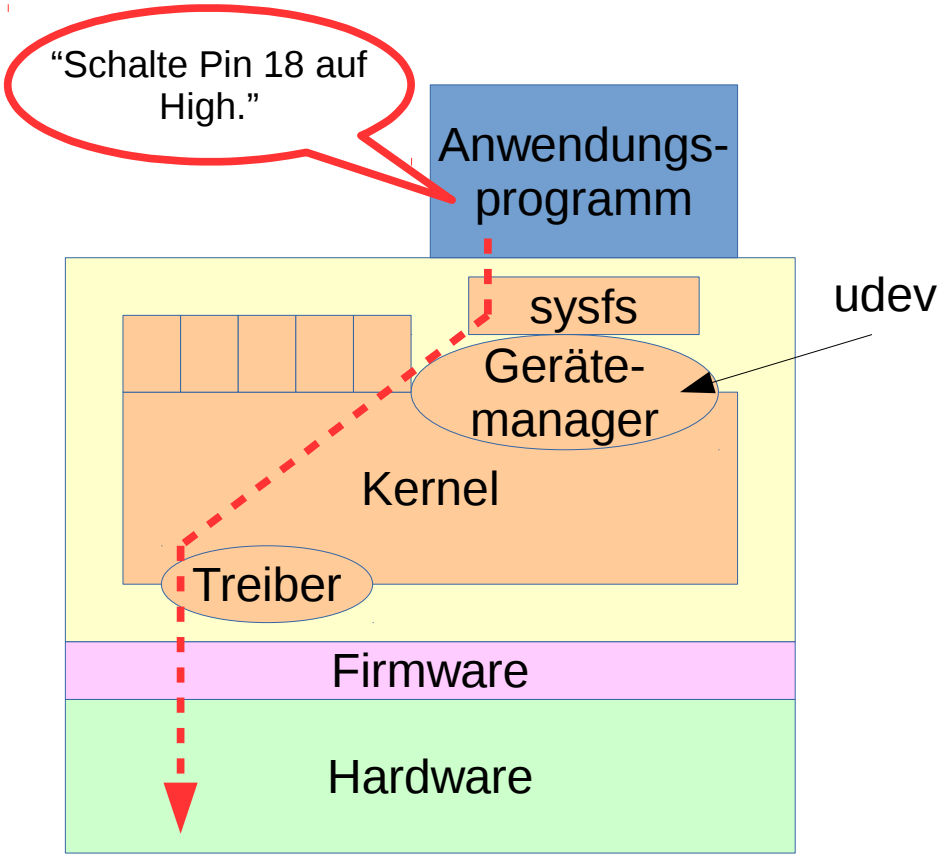


- Bei pull-down Konfiguration (default) der Eingänge muss gegen High (3.3 V) geschaltet werden.
- Bei pull-up Konfiguration der Eingänge muss gegen Masse (GND) geschaltet werden.

Das “Zeugs” zwischen Programm und dem GPIO-Pin



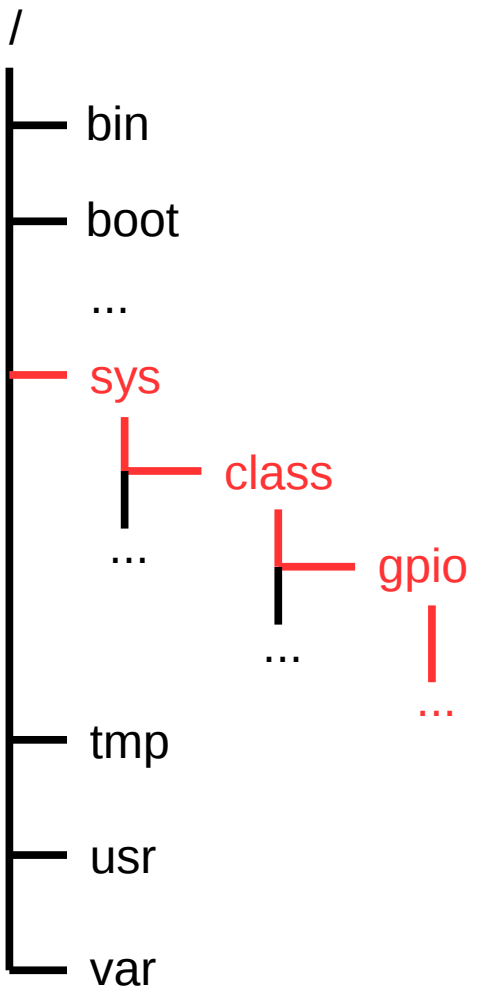
Klassischer µC (z.B. 8051)



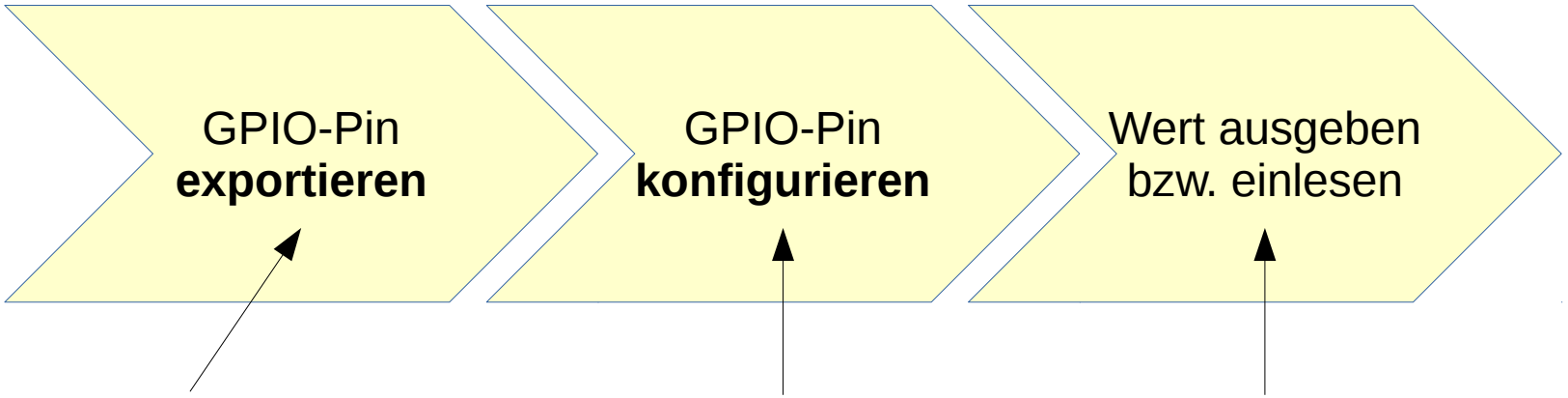
Raspberry Pi

Verwaltung durch das Betriebssystem

- Linux-Kernel unterstützt seit Version 2.6.21 Systeme mit GPIO-Schnittstelle
- Eine Schnittstelle zum Benutzer wird über *sysfs* bereitgestellt.
  Siehe auch:
  https://www.kernel.org/doc/Documentation/gpio/sysfs.txt
- “Everything is a file”



Inbetriebenahme eines GPIO-Pins über *sysfs*



BCM-GPIO-Nummer (z.B. 18 für GPIO18) muss in die Datei */sys/class/gpio/export* geschrieben werden.

Der Kernel erzeugt daraufhin in /sys/class/gpio ein weiteres Unterverzeichnis namens gpio18 mit weiteren Dateien

Im automatisch vom Kernel erstellten Unterverzeichnis befinden sich Dateien, über die man Einstellungen am Pin vornehmen kann.

- direction
- active_low
- edge

Bei Konfiguration des Pins als Ausgang schreibt man den auszugebenden Wert (entweder 0 oder 1) in die Datei /sys/class/gpio/gpio*XX*/value
Bei Konfiguration des Pins als Eingang kann aus dieser Datei der Zustand des Pins ausgelesen werden.

Erster GPIO Laborversuch (Sprachenunabhängig)

- Ein- und Ausschalten einer LED am GPIO
- Mögliche Lerninhalte (hardwarebezogen):
  - Berechnung des notwendigen Vorwiderstands
  - Schaltungstechnische Invertierung der Logik (Betrieb der LED sowohl high-aktiv als auch low-aktiv)
  - Praktischer Versuchsaufbau
  - Systematische Fehlersuche

Ansteuern einer LED über *sysfs* am konkreten Beispiel

- 1. Schritt: GPIO-Pin exportieren.
  echo “18” > /sys/class/gpio/export
- 2. Schritt: GPIO-Pin als Ausgang konfigurieren
  echo “out” > /sys/class/gpio/gpio18/direction
- 3. Schritt: Wert auf GPIO-Pin ausgeben
  echo “1” > /sys/class/gpio/gpio18/value

Hinweis: Ggf. sind für die o.g. Befehle root-Rechte notwendig: Der Befehlszeile sudo, gefolgt von einem Leerzeichen, voranstellen. Linux-Kenner können das Problem durch Ändern der udev-Konfiguration einstellen.

Blinkende LED mittels bash-Skript und Ansteuerung über *sysfs*

```
#!/bin/bash

echo "18" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio18/direction

while true;
do
  echo "1" > /sys/class/gpio/gpio18/value
  sleep 1
  echo "0" > /sys/class/gpio/gpio18/value
  sleep 1
done
```

- Das bash-Skript ausführbar machen: chmod +x skriptname
- Das Skript starten: ./skriptname

Hinweis: Ggf. sind diesen Aufruf root-Rechte notwendig: Der Befehlszeile sudo, gefolgt von einem Leerzeichen, voranstellen.

Einlesen des Zustands eines Pin über *sysfs* am konkreten Beispiel

- 1. Schritt: GPIO-Pin exportieren.
  echo “18” > /sys/class/gpio/export
- 2. Schritt: GPIO-Pin als **Eingang** konfigurieren
  echo “**in**” > /sys/class/gpio/gpio18/direction
- 3. Schritt: Wert von GPIO-Pin einlesen
  cat /sys/class/gpio/gpio18/value

Hinweis: Ggf. sind für die o.g. Befehle root-Rechte notwendig: Der Befehlszeile sudo, gefolgt von einem Leerzeichen, voranstellen. Linux-Kenner können das Problem durch Ändern der udev-Konfiguration einstellen.

Einfacher Inverter mittels bash-Skript und Ansteuerung über *sysfs*

```
#!/bin/bash

echo "18" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio18/direction
echo "17" > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio17/direction

while true;
do
  wert=`cat /sys/class/gpio/gpio17/value`
  if [ $wert -ne 0 ];
  then
    echo "0" > /sys/class/gpio/gpio18/value
  else
    echo "1" > /sys/class/gpio/gpio18/value
  fi
done
```

- LED (Ausgang) an GPIO 18
- Schalter (Eingang) an GPIO 17

Hinweis: Ggf. sind diesen Aufruf root-Rechte notwendig: Der Befehlszeile sudo, gefolgt von einem Leerzeichen, voranstellen.

Mögliche softwarebezogene Lerninhalte

- Abstraktion des Zugriffs auf Hardware durch das Betriebssystem
- Konkrete Unix/Linux-Grundlagen
  - “Everything is a file”
  - Absoluter vs. relativer Pfad
  - Navigieren in einer Verzeichnisstruktur
  - Umleiten des Standardausgabe-Streams in eine Datei
  - Zusammenfassung von Befehlen in einem bash-Skript
  - Logikfunktionen (NOT, AND, OR, ...)

Mögliche Nachteile der Verwendung von bash-Skripten im Unterricht

- Die bash gilt zwar als eine Turing-vollständige Sprache, wurde aber zur Automatisierung administrativer Abläufe und nicht zur Anwendungsprogrammierung entwickelt.
- (Hunds-)gemeine Syntaxregeln!
- Fehlermeldungen meist schwer zu verstehen und häufig irreführend
- Schnelle Demotivation, vor allem bei SuS
  - mit anderen Interessen
  - mit niederem “Durchhaltevermögen” und “Hang zur Bequemlichkeit”
  - mit (pathologischen) Konzentrationsstörungen

wiringPi

- Umfassende Bibliothek zum bequemen Zugriff auf die GPIO-Schnittstelle des Raspberry Pi
- OpenSource (GNU LGPLv3)
- Im Raspbian-Image bereits enthalten
- Stellt u.a. ein Kommandozeilen-Tool sowie Wrapper-Bibliotheken für eine Vielzahl von Programmiersprachen zur Verfügung
- Lässt sich durch existierende oder selbstgeschriebene Module zur Ansteuerung externer Hardware erweitern
- Webseite: http://wiringpi.com/

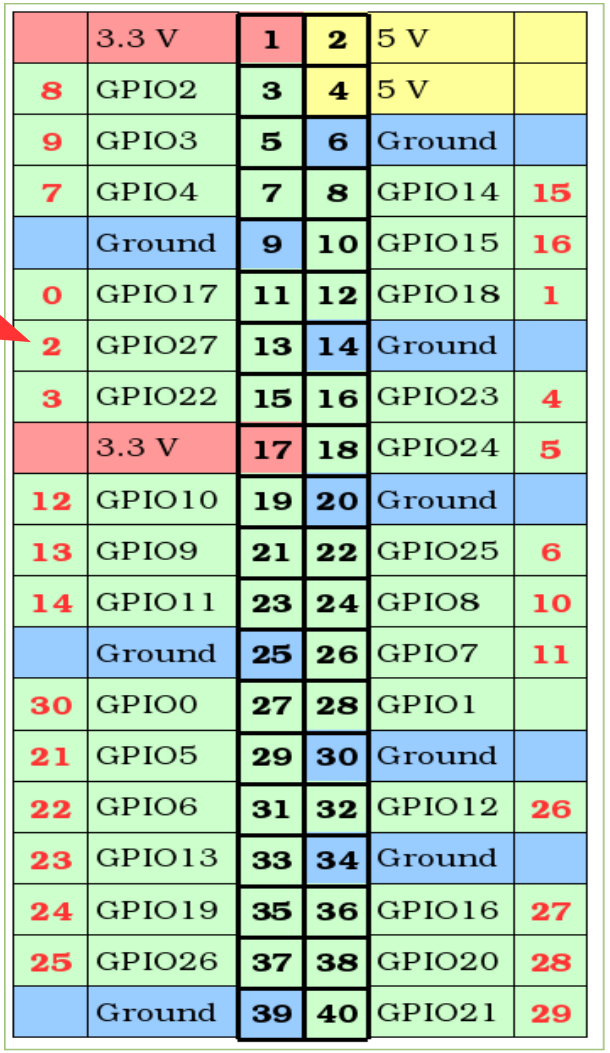
Eine Herausforderung für Schüler ...

Standard Nummerierung der Pins in wiringPi

Im Sprachgebrauch wird zwischen “wiringPi numbering” und “BCM numbering” a.k.a. “Broadcom numbering” unterschieden.

“Der physikalische Pin 5 ist der BCM GPIO 3 und damit der wiringPi Pin Nr. 9.”

<u>Aber:</u> wiringPi kann auf BCM-Nummerierung umgeschaltet werden.



	3.3 V	1	2	5 V	
8	GPIO2	3	4	5 V	
9	GPIO3	5	6	Ground	
7	GPIO4	7	8	GPIO14	15
	Ground	9	10	GPIO15	16
0	GPIO17	11	12	GPIO18	1
2	GPIO27	13	14	Ground	
3	GPIO22	15	16	GPIO23	4
	3.3 V	17	18	GPIO24	5
12	GPIO10	19	20	Ground	
13	GPIO9	21	22	GPIO25	6
14	GPIO11	23	24	GPIO8	10
	Ground	25	26	GPIO7	11
30	GPIO0	27	28	GPIO1	
21	GPIO5	29	30	Ground	
22	GPIO6	31	32	GPIO12	26
23	GPIO13	33	34	Ground	
24	GPIO19	35	36	GPIO16	27
25	GPIO26	37	38	GPIO20	28
	Ground	39	40	GPIO21	29

Kommandozeilen-Tool *gpio*

```
maierm@raspberrypi:~ $ gpio
Usage: gpio -v
       gpio -h
       gpio [-g|-1] ...
       gpio [-d] ...
       [-x extension:params] [[ -x ...]] ...
       gpio [-p] <read/write/wb> ...
       gpio <read/write/aread/awritewb/pwm/clock/mode> ...
       gpio <toggle/blink> <pin>
       gpio readall/reset
       gpio unexportall/exports
       gpio export/edge/unexport ...
       gpio wfi <pin> <mode>
       gpio drive <group> <value>
       gpio pwm-bal/pwm-ms
       gpio pwmr <range>
       gpio pwmc <divider>
       gpio load spi/i2c
       gpio unload spi/i2c
       gpio i2cd/i2cdetect
       gpio rbx/rbd
       gpio wb <value>
       gpio usbp high/low
       gpio gbr <channel>
       gpio gbw <channel> <value>
maierm@raspberrypi:~ $
```

Ansteuern einer LED über das *gpio*-Tool am konkreten Beispiel

- Exportieren und gleichzeitig als Ausgang konfigurieren
  gpio export 1 out
- Wert auf GPIO-Pin ausgeben
  gpio write 1 1
- Weitere Möglichkeiten
  gpio toggle 1
  oder
  gpio blink 1

Wenn's trotzdem nicht leuchtet und blinkt ...

- … liegt's immer an dem Menschen, der den Computer bedient. ;)
- Zur Eingrenzung des Fehlers bietet wiringPi weitere handliche Kommandos, z.B.
  gpio readall
- Details zur installierten Version
  gpio -v

Verwendung von wiringPi in C

- Library muss ins Programm includiert werden
  #include <wiringPi.h>
- Im Programm können nun Funktionen aus der wiringPi-Bibliothek verwendet werden. Eine Funktionsreferenz ist unter http://wiringpi.com/reference/ verfügbar.
- Wichtig: Beim Aufruf von gcc muss die Bibliothek für den Linker angegeben werden.
  gcc programm1.c -o programm1 -l wiringPi

Beispielprogramm in C

```
#include <wiringPi.h>

int main( int argc, char** argv )
{
  if( wiringPiSetup() == -1 )
    return 1;

  pinMode( 1, OUTPUT ); // BCM GPIO pin 18

  int i = 0;
  while( 1 )
  {
    digitalWrite( 1, 0 );
    delay( i );
    digitalWrite( 1, 1 );
    delay( i );
    i = ( i == 200 ) ? 0 : i + 10;
  }

  return 0 ;
}
```

Mögliche softwarebezogene Lerninhalte

- Einführung in die Programmiersprache C (unter entsprechendem Blackboxing)
- Verwendung von Bibliotheken
- Funktionsaufruf mit und ohne Argumente
- Verwendung von Konstanten
- Recherche in Software-Dokumentationen

Ansteuerung der GPIO-Schnittstelle in *Python*

- Python ist eine objektorientierte Skriptsprache
- Interpreter ist standardmäßig in Raspbian installiert.

```
maierm@raspberrypi:~ $ python
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hallo zusammen!"
Hallo zusammen!
>>> exit()
maierm@raspberrypi:~ $
```

- RPi.GPIO Modul ist standardmäßig bereits in Raspbian enthalten. (Raspbian-Packages python-rpi.gpio und python3-rpi.gpio)
- Webseite: https://pypi.python.org/pypi/RPi.GPIO
- Wiki: https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/

Beispielprogramm in Python

```
#!/usr/bin/python

import RPi.GPIO as GPIO
import time

GPIO.setmode( GPIO.BCM )
GPIO.setup( 18, GPIO.OUT )

while True:
   for outer in range( 1, 4 ):
      for inner in range( 1, 6 ):
         print "outer = ", outer, "; inner = ", inner
         GPIO.output( 18, GPIO.HIGH )
         time.sleep( 0.2 * outer )
         GPIO.output( 18, GPIO.LOW )
         time.sleep( 0.2 * outer )
```

Ausführen des Skriptes:

1. Möglichkeit: Ausführen durch expliziten Aufruf des Python-Interpreters:
python *skriptname*

2. Möglichkeit: Direktes Ausführen des Skripts als Kommando in der Shell, wobei der Python-Interpreter automatisch aufgerufen wird. Hierzu muss zunächst das Skript ausführbar gemacht werden
chmod u+x *skriptname*
Danach ist der Aufruf durch
./skriptname
möglich.

Mögliche Vorteile der Verwendung von Python im Unterricht

- OpenSource und kostenlos!
- Modern, objektorientiert und in der Industrie beliebt
- Syntax ist leichter zu erlernen als die von C/C++/C# oder Java
- High-level Sprache, d.h. sie erscheint der natürlichen Sprache relativ nahe. Python-Programme sind vereinfachtem Pseude-Code sehr ähnlich.
- Dynamische Typisierung (d.h. keine Deklaration von Variablen erforderlich)
- Plattformunabhängig
- Python enthält eine stattliche Anzahl an Standardbibliotheken (mehr als 300) für eine Vielzahl von Anwendungen

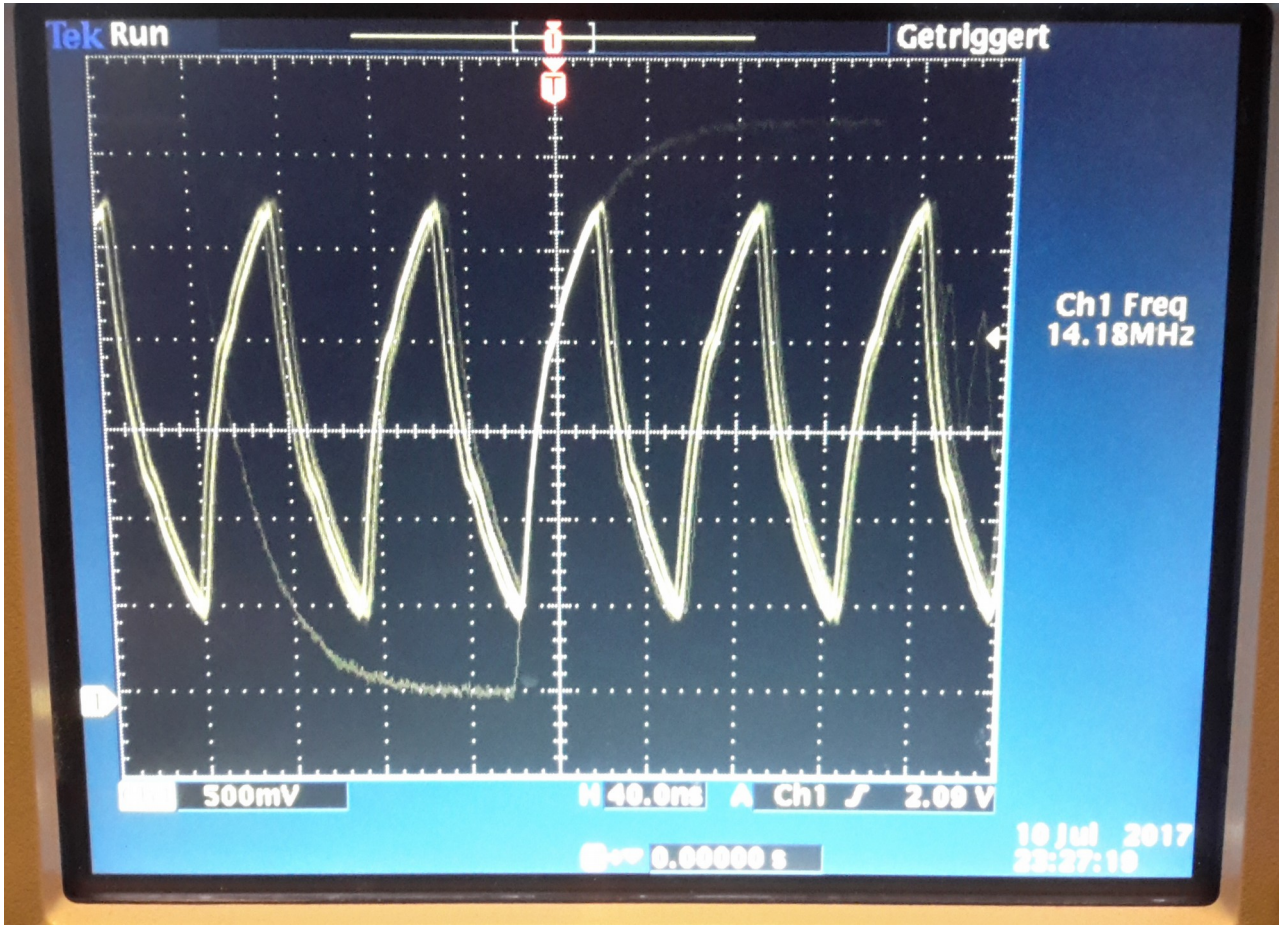
Vielleicht interessant aber sinnlos: Maximale Schaltfrequenzen

- Bei softwaremäßiger Ansteuerung der Pins kein Echtzeitcharakter -> starker Jitter -> nicht deterministisch!
- Frequenzen wurden empirisch an kaum belastetem System (load average 0.35) ermittelt

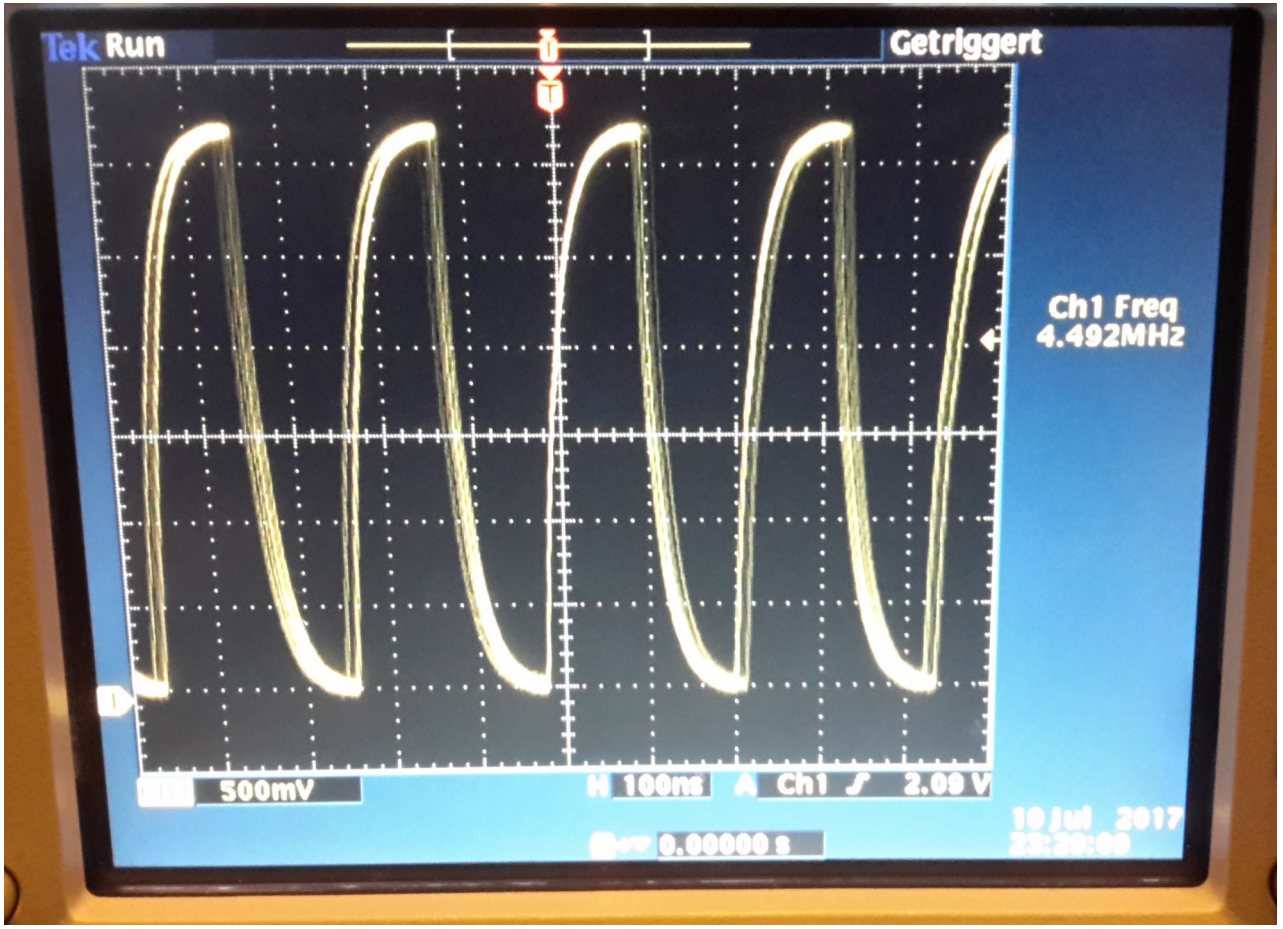
Ansteuerung	**max. Schaltfrequenz**
gpio-Tool mit bash-Skript	ca. 125 Hz
sysfs über bash-Skript	ca. 5.9 kHz
Python mit RPi.GPIO library	ca. 320 kHz
C mit wiringPi	ca. 13.7 MHz (*)
C mit wiringPi m. Verzögerung	ca. 4.5 MHz (**)
sym. Rechteck m. Hardware-PWM	ca. 4.8 MHz

(*) Signalflanken extrem verwaschen
(**) Verzögerung durch leere for-Schleifen mit 10 Durchläufen

Maximales Toggeln eines Pins mit wiringPi in C

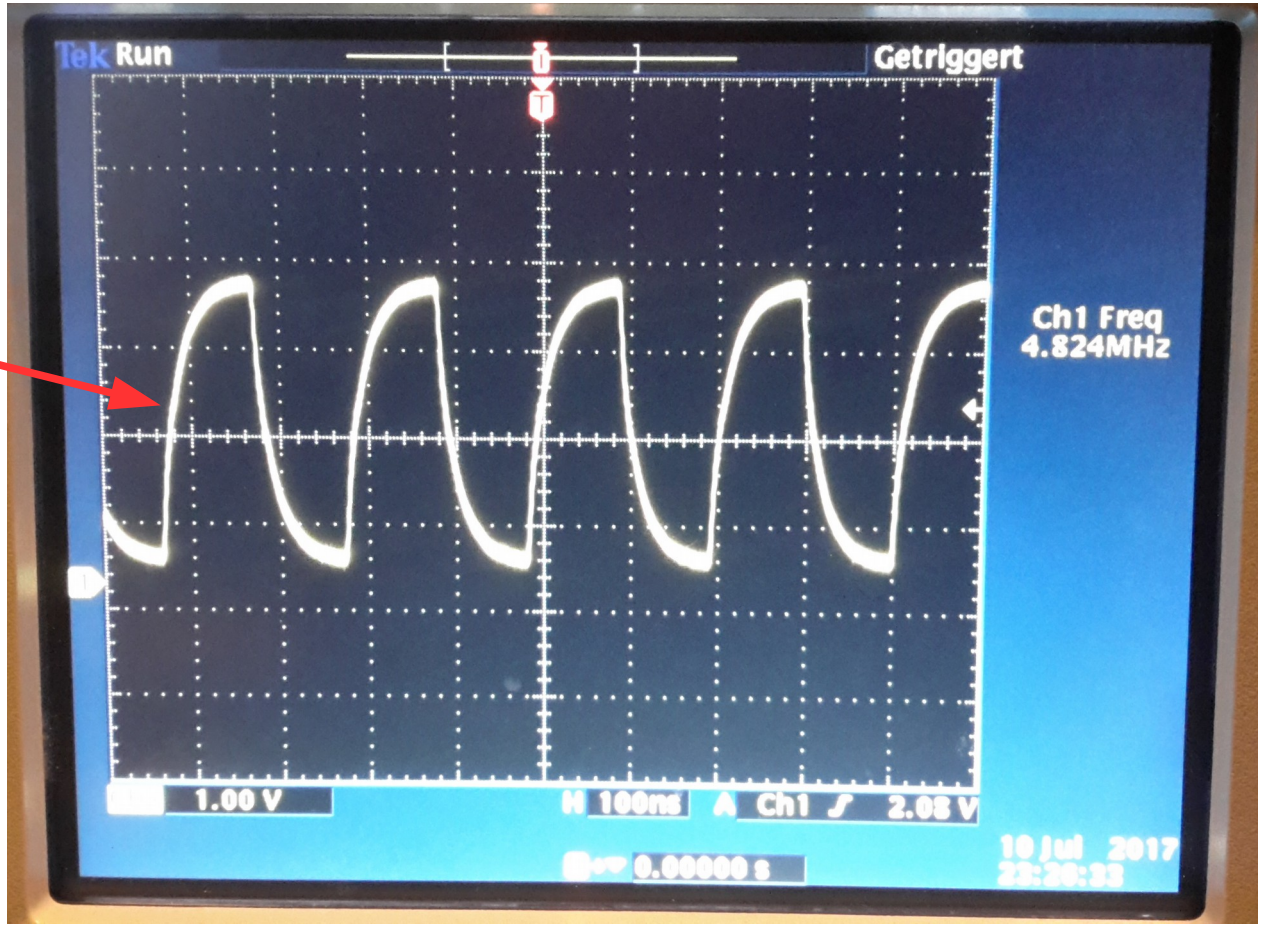


Toggeln eines Pins mit wiringPi in C mit Verzögerungsschleife



Hardware-PWM mit max. Frequenz und 50% Tastgrad

Jitterfrei, da durch hardware-generiert



GPIO-Pins mit speziellen Funktionen

- Bestimmte Pins der GPIO-Schnittstelle können für besondere Funktionen verwendet werden.
  - Hardware PWM (BCM-Pins 12, 18, 13, 19)
  - Hardware-Clock Generator (BCM-Pins 4, 5, 6)
  - I²C-Bus
  - SPI-Bus
  - UART
- Belegung siehe hier: https://pinout.xyz

Hardware PWM

- Der BCM-Chip bietet zwei voneinander unabhängige PWM-Kanäle (PWM0 und PWM1)
- Beide PWM-Kanäle werden aus der selben internen Taktquelle (19.2 MHz) betrieben, wobei die Frequenz der PWM durch einen einstellbaren Teiler beeinflusst werden kann, der den irreführenden Namen PWM Clock (PWMC) trägt.
- Grundsätzlich kann die PWM in zwei Modi betrieben werden:
  - PWM im “mark space mode” (PWM MS)
    Traditionelle PWM, mit einer vom Tastgrad unabhängigen Frequenz. Dieser Modus stellt den häufigsten Anwendungsfall dar (z.B. Ansteuerung von Servos).
  - PWM im “balanced mode” (PWM BAL)
    Die Frequenz kann (und wird) sich in Abhängigkeit des PWM Werts ändern. Algorithmus wird im BCM283x-Datenblatt beschrieben.

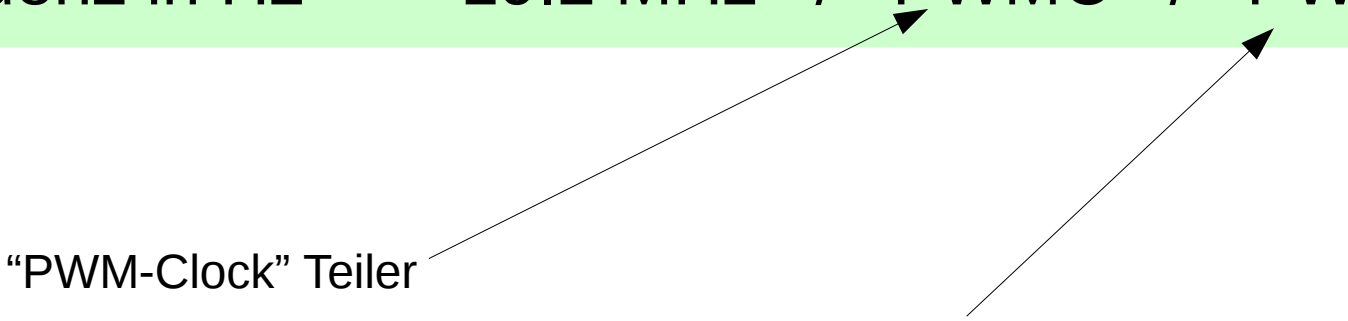
PWM Clock Teiler (PWMC)

PWMC	PWM-Frequenz
1	4.6875 kHz
2	9.6 MHz
4	4.8 MHz
8	2.4 MHz
16	1.2 MHz
32	600 kHz
64	300 kHz
128	150 kHz
256	75 kHz
512	37.5 kHz
1024	18.75 kHz
2048	9.375 kHz

- Ausgangsfrequenz ist immer 19.2 MHz
- Der zulässige Wertebereich für den PWMC reicht von 1 (inklusive) bis 4095 (inklusive)
- Der PWMC-Wert 1 stellt einen Teilerfaktor von 4096 dar.

Berechnung der PWM-Frequenz im “mark space” Modus

PWM-Frequenz in Hz = 19.2 MHz / PWMC / PWMR



Hardware-PWM mit *gpio* Tool auf der Kommandozeile

```
gpio mode 1 pwm
gpio pwm-ms
gpio pwmc 1920
gpio pwmr 200
gpio pwm 1 20
```

GPIO 18 (wiringPi Pin 1) auf Hardware-PWM umstellen.

PWM soll im “mark space”-Modus betrieben werden.

PWM-Takt beträgt
19.2 MHz / 1920 = 10 KHz

PWM-Range (PWMR): Die 10 kHz werden durch 200 Einheiten geteilt, d.h. die PWM-Frequenz beträgt
10 KHz / 200 = 50 Hz

Das Ausgangssignal wird für 20 Einheiten auf High geschaltet. Bei einer PWM-Range (PWMR) von 200 Einheiten ergibt das einen Tastgrad von 10%.

Hardware-PWM mit wiringPi in C

```
#include <wiringPi.h>

int main( int argc, char** argv )
{

  if( wiringPiSetup() == -1 )
    return 1;

  pinMode( 1, PWM_OUTPUT ); // BCM GPIO pin 18
  pwmSetMode( PWM_MODE_MS );
  pwmSetClock( 1920 );
  pwmSetRange( 200 );
  pwmWrite( 1, 10 );

  return 0 ;

}
```

Achtung:

C-Programm benötigt root-Rechte, ansonsten stürzt der Raspberry Pi ab!

Ausführen mit:
sudo ./programmname

Einsatz der Hardware-PWM des Raspberry Pi im Unterricht

- Mögliche Lerninhalte
  - Grundprinzip der Leistungssteuerung mittels PWM
  - Begriffe Frequenz, Periodendauer, Tastgrad
  - Rechnen mit 10er-Potenzen
  - Übungen am Oszilloskop
  - Abgrenzung zur Software-PWM (Vorteile, Nachteile)
  - Datenblatt-Recherche (in diesem Fall wohl eher für Technikerschüler angemessen)
- Mögliche Hürden für die Schüler
  - Relativ komplexe und teilweise “schwammige” Dokumentation
  - Eigenheiten bei den zulässigen Wertebereichen der PWM-Parameter
  - Raspberry Pi-Absturz bei fehlenden root-Rechten (wiringPi, C)

Beispiele für externe Erweiterungen für den hardwarenahen Unterricht

- Bausteine der 74HC...-Familie (Log. Grundfunktionen, Schieberegister, ...)
- NE555 Timer
- L293 H-Brücke zur Steuerung von DC-Motoren
- Div. I²C-Sensoren, z.B. ADXL345 Beschleunigungssensor
- Fertige Expansionboards, siehe z.B. http://elinux.org/RPi_Expansion_Boards

Zum Abschluss: Workshop / Ideen

- Pulsierende LED
- Nachbildung eines Logikbausteins, z.B. XOR
- Digitaler Würfel
- Für C-Cracks: Ausgabe der IP-Adresse in Morsecode auf einer angeschlossenen LED