

# System-Programmierung

## 14: Weitere Arten von I/O

CC BY-SA, Thomas Amberg, FHNW  
(soweit nicht anders vermerkt)



### Ablauf heute

⅓ Vorlesung,  
⅔ Hands-on,  
Feedback.

Slides, Code & Hands-on: [tmb.gr/syspr-14](http://tmb.gr/syspr-14)



### Weitere Arten von I/O

Die folgenden Arten von Input und Output gehören nicht strikt zu System-Programmierung, sind aber im Zusammenhang mit interaktiven "embedded" Linux Anwendungen von Bedeutung.

Dieses Kapitel basiert auf der Raspberry Pi Hardware, die Konzepte sind aber allgemein gültig. Bibliotheken die Raspberry Pi spezifisch sind, werden nicht geprüft.



### GPIO

*GPIO*, General Purpose Input/Output sind unbesetzte CPU Pins, welche per Programm wahlweise als Signal Input oder Output verwendet werden können.

Allgemein gibt es analoge und digitale GPIO Pins, der Raspberry Pi bietet aber nur digitale Inputs/Outputs.

Das *Pinout* ist Hardware-abhängig, die Ansteuerung der Pins erfordert eine nicht-Standard Bibliothek.



### Digital Input/Output

Ein GPIO Pin der als digitaler *Input* konfiguriert ist, hat entweder den Wert 1 oder 0 (*HIGH* oder *LOW*).

Ein GPIO Pin der als digitaler *Output* konfiguriert ist, wird auf den Wert 1 oder 0 gesetzt (*HIGH* od. *LOW*).

Die Spannung für HIGH/LOW hängt von der Hardware ab, Raspberry Pi verwendet 3.3V Logik.



### WiringPi

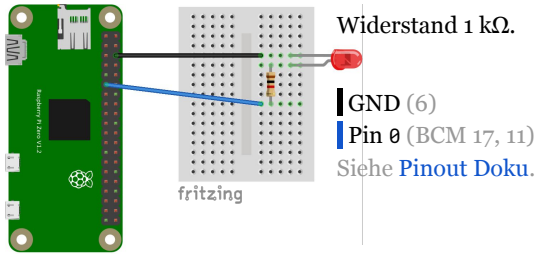
*WiringPi* ist eine einfache, Arduino-ähnliche GPIO Bibliothek in C für alle Raspberry Pi Versionen, d.h. für die SoC *BCM2835*, *BCM2836* und *BCM2837*.

Dokumentation gibt es unter <http://wiringpi.com/>, installiert wird die Library auf Raspberry Pi mit:

```
$ git clone git://git.drogon.net/wiringPi
$ cd wiringPi
$ ./build
```



### Beispiel: LED anschliessen



n|w

### Digitaler Output

[blink.c](#)

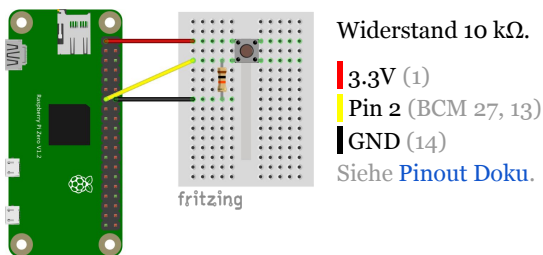
Digitalen Output auf den GPIO Pin 0 schreiben:

```
#include <wiringPi.h>
```

```
int main(void) {  
    wiringPiSetup();  
    pinMode(0, OUTPUT);  
    while(1) {  
        digitalWrite(0, HIGH); delay(500);  
        digitalWrite(0, LOW); delay(500);  
    }  
}
```

n|w

### Beispiel: Schalter anschliessen



n|w

### Digitaler Input

[input.c](#)

Digitalen Input von GPIO Pin 2 lesen:

```
#include <wiringPi.h> ...
```

```
int main(void) {  
    wiringPiSetup();  
    pinMode(2, INPUT);  
    while(1) {  
        int value = digitalRead(2);  
        printf("%d\n", value);  
    }  
}
```

n|w

### Hands-on, 30': GPIO

[switch.!c](#)

Diese Hands-on Übung basiert auf dem [GPIO Pinout](#) des Raspberry Pi und kann gut zu zweit gelöst werden.

Schreiben Sie ein Programm *my\_switch.c* das mittels GPIO den Zustand eines Buttons liest, und damit eine LED ein- bzw. ausschaltet.

Während dem Aufbau der Schaltung sollte das Gerät vom Strom getrennt sein, zum Schutz der Elektronik.

n|w

### UART

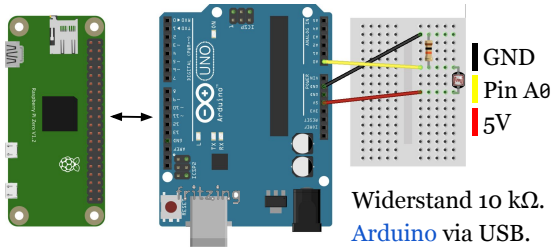
UART, Universal asynchronous receiver/transmitter, ist ein weit verbreitetes "Wire-Protokoll" für serielle Kommunikation.

Eine UART Verbindung kann physisch entweder über USB oder direkt über GPIO Pins erstellt werden.

Die Dokumentation zu Raspberry Pi UART ist [hier](#), WiringPi enthält eine einfache [Serial Library](#).

n|w

### Bsp.: Analoger Lichtsensor anschliessen



### Beispiel: Analoger Input via Arduino

Arduino ist ein Mikrocontroller mit Analog In-/Out.

Arduino via USB mit Computer verbinden.

In der [Arduino IDE](#) ist das Programm  
Examples > Basics > AnalogReadSerial

Nach dem Upload läuft das Programm "stand-alone"  
auf dem Arduino und schreibt Output auf USB Serial.

Raspberry Pi kann Output via USB Serial lesen.

### Analoger Input via UART

[uart.c](#)

Analogen Input (z.B. von Arduino) via UART lesen:

```
#include <wiringSerial.h> ...  
  
int main(void) {  
    int fd = serialOpen("/dev/serial0", 9600); *  
    int ch = serialGetchar(fd);  
    while (ch != -1) {  
        putchar(ch);  
        ch = serialGetchar(fd);  
    }  
} // *) oder "/dev/ttyAMA0", je nach Pi
```

### Hands-on, 45': GPS via UART

[gps.!c](#)

Diese Hands-on Übung basiert auf dem [UART Pinout](#)  
des Raspberry Pi, und kann in Gruppen gelöst werden.

Schreiben Sie ein Programm *my\_gps.c* das via UART/  
AT Commands ein GPS Modul anspricht und ausliest.

Als Hardware dient das [ublox PAM 7Q](#) GPS Modul.

**Achtung:** Verkabeln nur mit ausgestecktem Raspberry Pi,  
VCC des GPS Moduls nur mit 3V3 verbinden, nicht 5V.

### Selbststudium, 3h: TTY

Lesen Sie diese Einführung zum TTY Subsystem:

<http://linusakesson.net/programming/tty/index.php>

### Feedback?

Gerne im [Slack](#) oder an [thomas.amberg@fhnw.ch](mailto:thomas.amberg@fhnw.ch)

Programmierfragen am besten schriftlich.

Sprechstunde auf Voranmeldung.

Slides, Code & Hands-on: [tmb.gr/syspr-14](https://tmb.gr/syspr-14)

