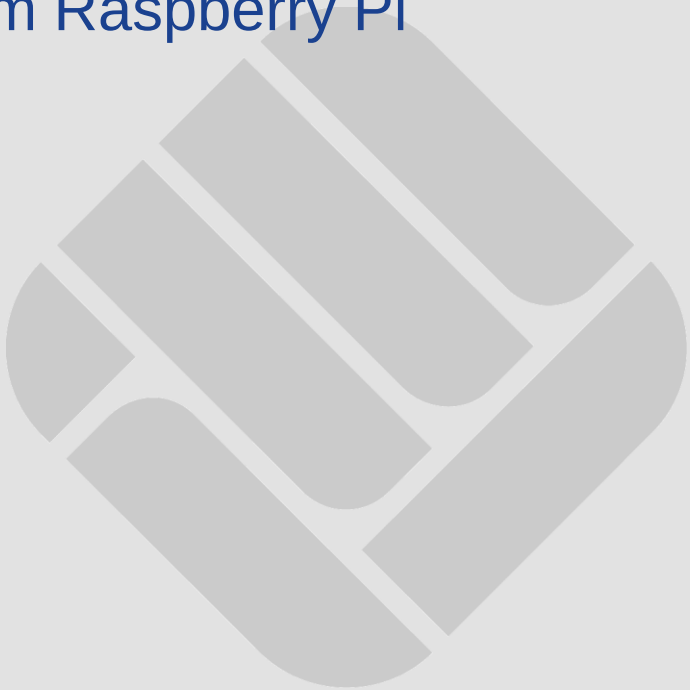
Fachhochschule
Münster University of
Applied Sciences

Embedded Software

Auslesen eines Smart-Meter mit dem Raspberry Pi



Stefan Bücker, Tim Hebbeler | Fachbereich für Elektrotechnik und Informatik FB2

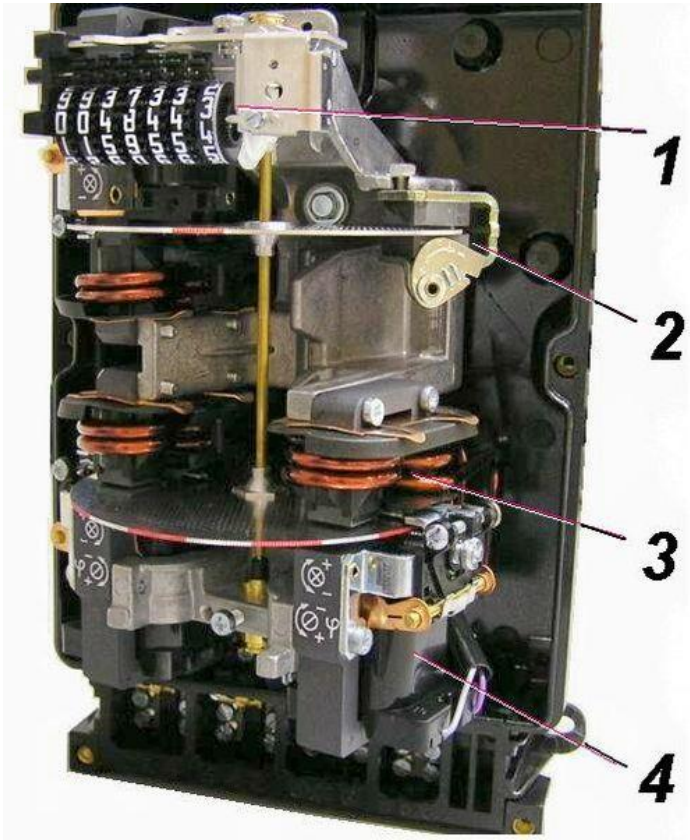
- Vorstellen des EasyMeter / SmartMeter
- Grundlagen des UART / Serielle Schnittstelle
  - Einsatzgebiete von UART
- Protokoll des EasyMeter erläutern
- Einsatz des UART auf dem Raspberry Pi
  - Erstellen einer Seriellen Verbindung
- Aufgabenstellung
- Lösungsansatz
  - Kurzeinführung pthreads Prozesse vs. Threads

- Lösung

Stromzähler / elektromechanischer Stromzähler

- Eigentlich „Energiezähler“
- Messeinheit Kilowattstunde (kWh)
- Ferraris-Zähler
- spezielle Form des Asynchronmotor

Stromzähler / elektromechanischer Stromzähler 2



1. Rollenzählwerk
2. justierbare Wirbelstrombremse (Dauermagnet)
3. eine der drei Stromspulen
4. eine der drei Spannungsspulen (die dritte versteckt sich links hinter dem Zähler)

- drehbar gelagerten Aluminiumscheibe
- Zwei Erregerspulen
  1. wenige Windungen => Strompfad
  2. hoher Impedanz => Spannungspfad
- Spulen erzeugen ein magnetisches Drehfeld
- Drehmoment ist proportional zum Produkt aus Strom und Spannung.
- Es wird keine Blindleistung gemessen.

„intelligenter“ Zähler / Smart Meter

Definition laut § 21d EnWG:

- Erfassung von
  - elektrischer Energie
  - tatsächlichem Energieverbrauch
  - tatsächlicher Nutzungszeit

- Eingebunden in ein Kommunikationsnetz

European Smart Metering Alliance (ESMA):

- Nur „smart“ mit Mikrocontroller

Smart-Meter Vorteile

- Vorteile:
  - Endkunde:
    - Energieverbrauch über den Tag ist sichtbar
    - Möglichkeit des Kunden stromintensive Gerätschaften in günstigeren Tarifzeiträumen einsetzen zu können.
    - Kein manuelles Ablesen des Stromzählers mehr nötig, sofern dieser über eine Kommunikationseinheit zum Energieversorger verfügt.
  - Energieversorger:
    - Variable Tarifgestaltung über den Tag.
    - vorhandene Kraftwerkinfrastruktur kann besser ausgenutzt werden und Investitionen für den Spitzenlastausbau wird vermieden.

Smart-Meter Nachteile

- Nachteile
  - Endkunden
    - Ausspähung der Kundengewohnheiten über den Energieverbauch durch Kommunikation des Smart-Meters mit dem Energieerzeuger
    - Anschaffung und Betrieb des Smart-Meters ist teurer als bei **Ferraris-Zähler**
    - Gefahr undurchsichtiger Preispolitik der Energieerzeuger.
  - Energieerzeuger
    - Aufbau einer Infrastruktur zum Auslesen alle Kunden-Smart-Meter.

Unserer Smart-Meter von der Firma Easymeter

- Basiszähler Q3D
  - 3-phasiger / 4 Leiter- Zähler
  - Infoanzeige in Watt
  - Unidirektionale D0-MSB-Schnittstelle (DIN EN 625056-61)(optisch)
    - Protokoll mit OBIS Kennzahlen
  - Erweiterungen:
    - Kommunikation über Ethernet, GPRS / GSM, Power Line Communication, Wireless M-Bus
    - Funk- und Tonrundsteuerung
    - In-Haus Kommunikation

Easymeter Q3DA1004

- Unser Model Q3DA1004
  - DIN Gehäuse, LC- Display, Eintarif (Gesamtenergie) Genauigkeitsklasse A =>2%
  - 5/60 A (DIN-Anschlussklemmen Ø 6,5mm)
  - Direktmessung
  - Bezugszähler mit Rücklaufsperre

Easymeter Q3D

- Interner Aufbau
  - Strommessung über Shuntwiderstand
  - Hochauflösender ADC wandelt die Werte um.
  - Mikrocontroller berechnet die Energiewerte und speichert diese in einem nichtflüchtigen Speicher.
- Optische D0-Schnittstelle (OBIS-Protokoll)
  - Alles 2 sec wird ein Datenpaket gesendet
  - Zeichenweise Übertragung der Daten

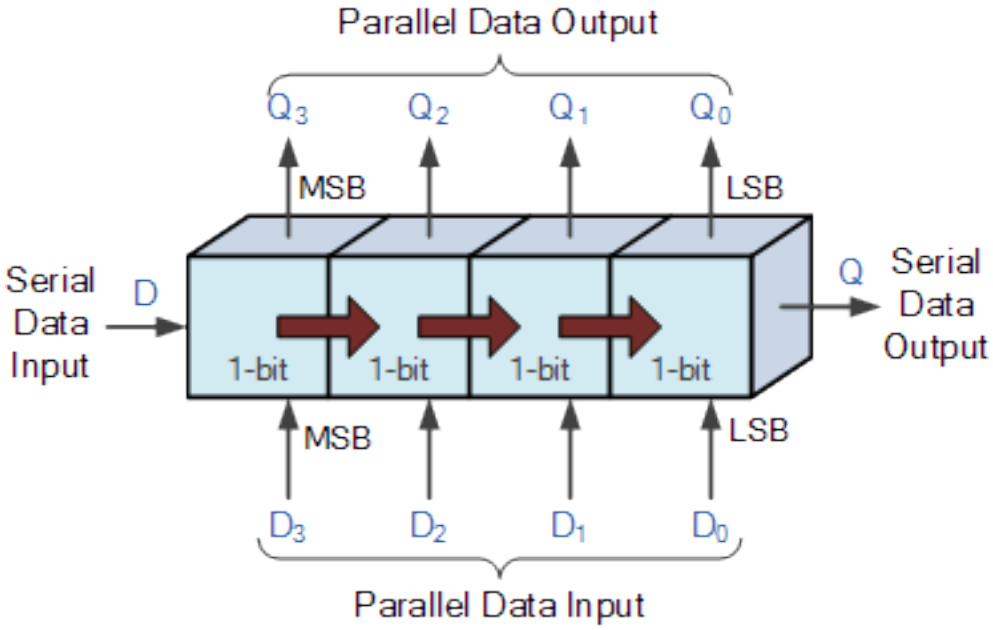
start	2^0	2^1	2^2	2^3	2^4	2^5	2^6	parity	stop

- =>UART

Grundlagen des UART

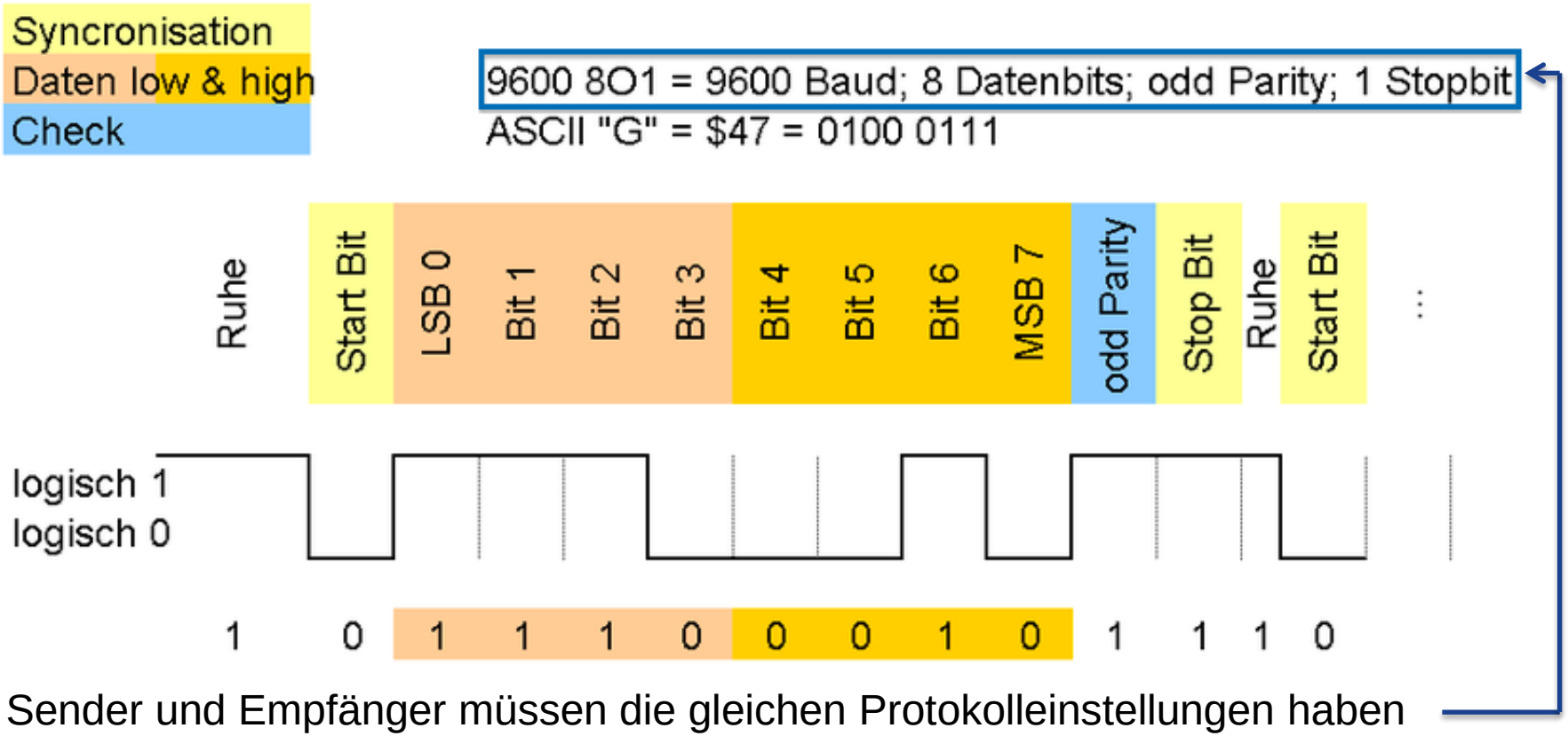
- **UART** steht für **U**niversal **A**synchronous **R**eceiver/**T**ransmitter

- Schnittstelle von der parallelen zur seriellen Übertragung und umgekehrt



Grundlagen des UART
Rahmen

- Daten werden in einem Rahmen zwischen Start und Stop-Bit übertragen
- Ohne Kommunikation ist der Pegel auf der Leitung „High“



Sender und Empfänger müssen die gleichen Protokolleinstellungen haben

Grundlagen UART
Einsatzgebiete

- Kommunikation mit dem PC z.B. RS-232
- Datenübertragung per IR
- Bootloader

Protokoll des EasyMeter

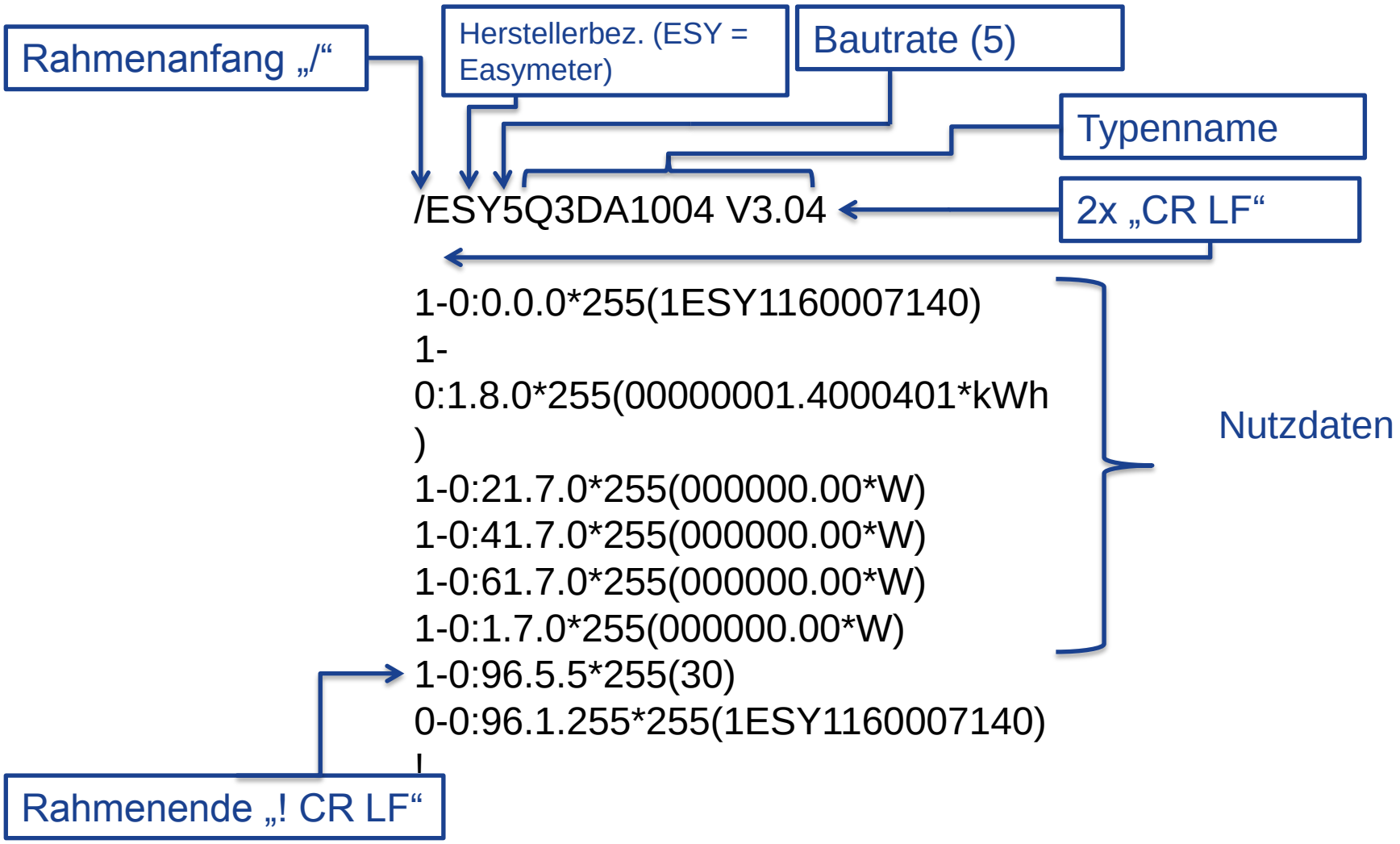
start	2^0	2^1	2^2	2^3	2^4	2^5	2^6	parity	stop

- UART Einstellungen
  - Baudrate = 9600
  - 1 start bit
  - 7 data bits
  - 1 parity bit (even)
  - 1 stop bit

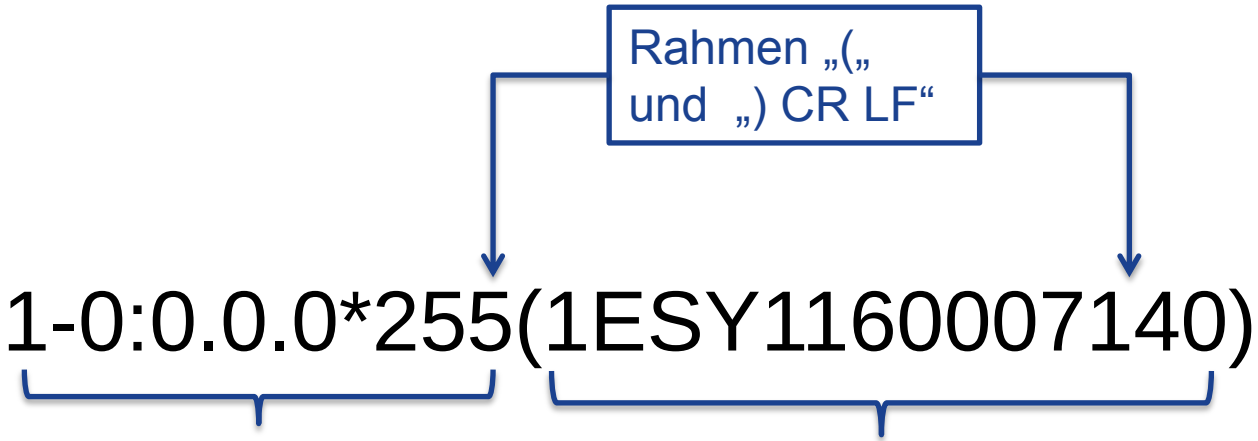
Protokoll des EasyMeter 2

```
/ESY5Q3DA1004 V3.04

1-0:0.0.0*255(1ESY1160007140)
1-0:1.8.0*255(00000001.4000401*kWh)
1-0:21.7.0*255(000000.00*W)
1-0:41.7.0*255(000000.00*W)
1-0:61.7.0*255(000000.00*W)
1-0:1.7.0*255(000000.00*W)
1-0:96.5.5*255(30)
0-0:96.1.255*255(1ESY1160007140)
!
```

D0 Schnittstelle des EasyMeter



D0 Schnittstelle des EasyMeter 2



OBIS ("Object Identification System") Kennzahlen

A-B:C.D.E*F

Nutzdaten

A = Medium
B = Kanal
C = Messgröße
D = Messart
E = Tarif
F = Vorwert

DIN EN 62056-61

D0 Schnittstelle des EasyMeter 3

Medium (A)	Kanal (B)	Messgröße (C)	Messart (D)	Tarif (E)
1 Elektrizität	Kanal 0 bis 64	1 ∑ Li Wirkleistung + 2 ∑ Li Wirkleistung - 3 ∑ Li Blindleistung + 4 ∑ Li Blindleistung - 5 ∑ Li Blindleistung Q I 6 ∑ Li Blindleistung Q II 7 ∑ Li Blindleistung Q III 8 ∑ Li Blindleistung Q IV	6 Maximum 8 Zeitintegral 1 9 Zeitintegral 2 29 Zeitintegral 5	0 Total 1 Tarif 1 2 Tarif 2 3 Tarif 3 4 Tarif 4 5 Tarif 5 ... ... 9 Tarif 9

Ziel / Bedeutung	OBIS	Kommentar
L1+ Active Power (momentane Leistung P1)	1-0:21.7.255*255 (instantaneous)	Momentane Leistung –6 Stellen +2 Nachkommastelen in W mit Vorzeichen (-123456,12*W)
Σ Li+ Active Power (momentane Summe der Leistung)	1-0:1.7.255*255	Momentane Summe der Leistung – 6 Stellen +2 Nachkommastelen in W mit Vorzeichen (-123456,12*W)
Statusinformation	1-0:96.5.5*255	Das Statuswort wird als ein Byte definiert und in hexadezimaler Darstellung übertragen. Es gilt folgende Zuordnung: Bit[7] – MSB, 0=Leerlauf, 1=oberhalb Anlauf Bit[6] – beim Phasenausfall L1 wird gesetzt Bit[5] – beim Phasenausfall L2 wird gesetzt Bit[4] – beim Phasenausfall L3 wird gesetzt Bit[3:2] – reserviert, immer 0 Bit[1] – ‚1' das Telegramm wird immer synchron im festen Zeitraster ausgegeben Bit[0] – ‚0' kein Fehler, ‚1' – Fehler

Einsatz des UART auf dem Raspberry Pi

- Frei machen des UART
  - /boot/cmdline.txt bearbeiten
    - dwc_otg.lpm_enable=0 ~~console=ttyAMA0,115200 kgdboc=ttyAMA0,115200~~ console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
  - /etc/inittab bearbeiten
    - Zeile „ T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100“ mit „#“ auskommentieren
- UART testen
  - Senden: $ echo Text >> /dev/ttyAMA0
  - Empfangen: $ cat /dev/ttyAMA0 -s

Mit dem UART unter Linux arbeiten

- Open auf Device
  - int fd = open ("/dev/ttyAMA0", O_RDWR | O_NOCTTY);

- Termios

#include <termios.h>

#include <unistd.h>

…

struct termios options

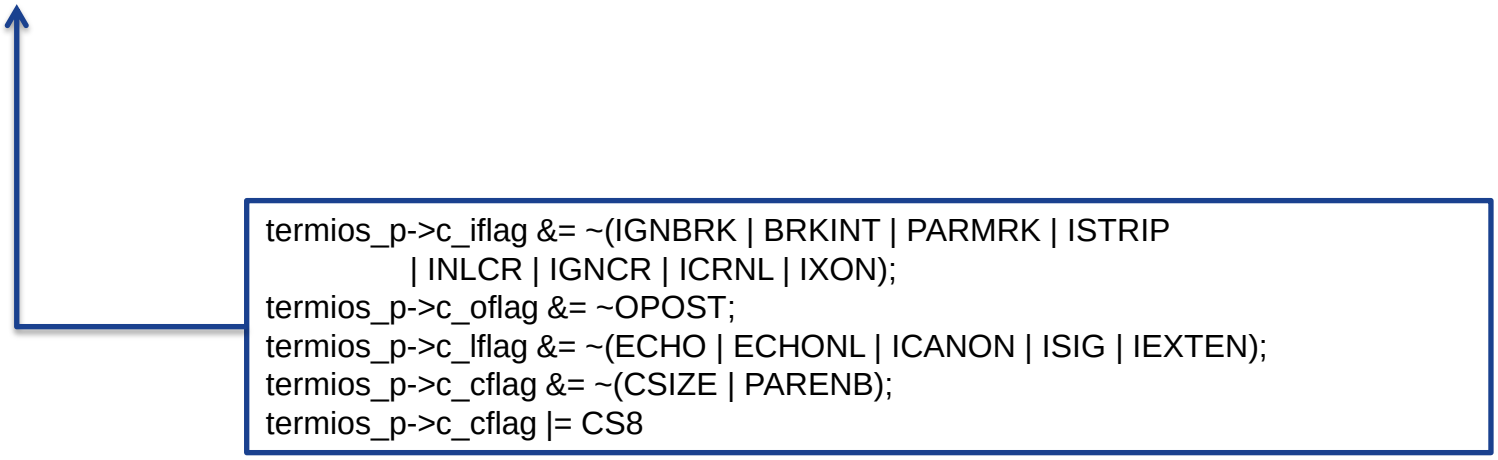
```
tcflag_t c_iflag;      /* input modes */
tcflag_t c_oflag;      /* output modes */
tcflag_t c_cflag;      /* control modes */
tcflag_t c_lflag;      /* local modes */
cc_t     c_cc[NCCS];   /* special characters */
```

Einsatz des UART auf dem Raspberry Pi

```
tcgetattr (fd, &options) ;  //holt die Parameter des Filedescriptor und speichert es in "options"

cfmakeraw   (&options) ;  //Setzt c_iflag,c_oflag,c_cflag und c_lflag zu raw mode
cfsetispeed (&options, myBaud) ; // setzt die Geschwindigkeit für RX
cfsetospeed (&options, myBaud) ; // setzt die Geschwindigkeit für TX

options.c_cflag |= (CLOCAL /* Ignoriert Modem Control Lines*/| CREAD /* Aktiviert RX*/) ;
options.c_cflag |= PARENB ;  // Aktiviert das Parity Bit (gerade)
options.c_cflag &= ~CSTOPB ;  // Deaktiviert das 2.Stopbit
options.c_cflag &= ~CSIZE ;      // Bitmaske für CS5, CS6, CS7 ,CS8
options.c_cflag |= CS7 ;   // Character Size (Größe eines Datenwortes)

options.c_lflag &= ~(ICANON /*kein canonischer Modus*/| ECHO | ECHOE | ISIG) ;
options.c_oflag &= ~OPOST ;

options.c_cc [VMIN]  =   0 ;
options.c_cc [VTIME] = 10 ;       // Ten seconds (100 deciseconds)

tcsetattr (fd, TCSANOW | TCSAFLUSH, &options) ;
```

Termios Linux

```
//holt die Parameter des Filedescriptor und speichert es in "options"
tcgetattr (fd, &options) ;

//Setzt c_iflag,c_oflag,c_cflag und c_lflag zu raw mode
cfmakeraw (&options) ;
```



Termios Linux 2

```
cfsetispeed (&options, myBaud) ; // setzt die Geschwindigkeit für RX
cfsetospeed (&options, myBaud) ; // setzt die Geschwindigkeit für TX

options.c_cflag |= (CLOCAL /* Ignoriert Modem Control Lines*/| CREAD /* Aktiviert RX*/) ;
options.c_cflag |= PARENB ;  // Aktiviert das Parity Bit (gerade)
options.c_cflag &= ~CSTOPB ;     // Deaktiviert das 2.Stopbit
options.c_cflag &= ~CSIZE ; // Bitmaske für CS5, CS6, CS7 ,CS8
options.c_cflag |= CS7 ;// Character Size (Größe eines Datenwortes)

options.c_lflag &= ~(ICANON /*kein canonischer Modus*/| ECHO /*deaktiviere Echo Modus
*/| ECHOE | ISIG) ;
options.c_oflag &= ~OPOST ;

tcsetattr (fd, TCSANOW /*Änderung wird sofort aktiv*/ | TCSAFLUSH /*Im Puffer befindliche
Zeichen werden gelöscht*/, &options);
```

Lesen von einem File

```
ssize_t read(int fd, void *buf, size_t count);
```

```
int sscanf ( const char * s, const char * format, ...);
```

Aufgabenstellung

1. Testen Sie die Funktion Ihrer seriellen Schnittstelle, sodass Sie Daten senden und empfangen können.

2. Nachdem Sie sich vergewissert haben das Ihre serielle Schnittstelle funktioniert, bauen Sie die Schaltungen zum senden und empfangen auf. Schreiben Sie anschließend ein C Programm, dass die Beispieldaten sendet und empfängt.

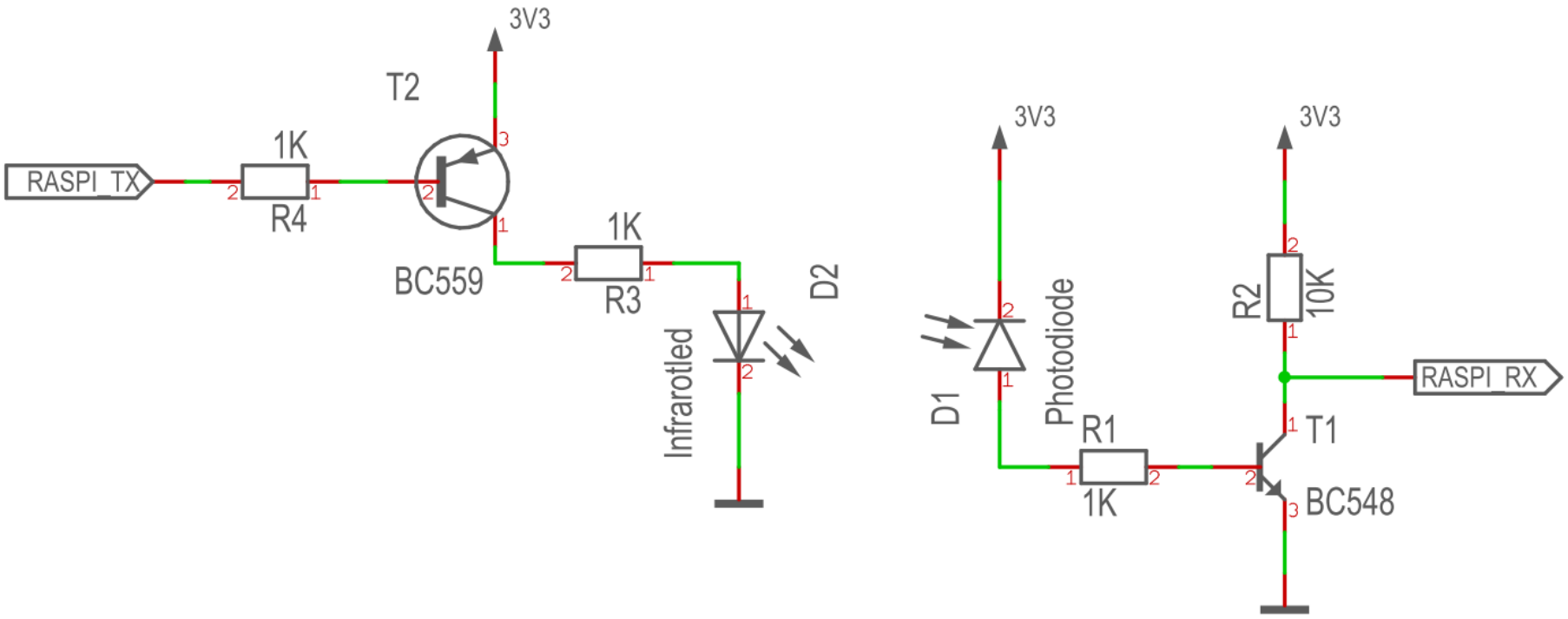
3. Schreiben Sie ein C-Programm welches die Empfangenen Daten in eine Datei speichert.

4. Verwenden Sie nun das Easymeter als Quelle für Ihre serielle Schnittstelle.

Hinweise zur Aufgabenstellung
zu Aufgabe 2

- Bauteile für die Schaltung:
  - Sender: - 1x PNP-Transistor BC559
    - 2x 1kΩ Widerstände
    - 1x Sendediode IR OPE 5685 850nm
  - Empfänger: - 1x IR Empfangsdiode OSRAM 850nm
    - 1kΩ Widerstand Basisvorwiderstand
    - 10kΩ Kollektorwiderstand
    - 1x NPN-Transistor BC548
  - Die Schaltung soll so aufgebaut werden, das jeweils der Empfänger und der Sender das Signal invertieren.

Schaltplan


3V3
T2
RASPI_TX
1K
R4
BC559
1K
R3
D2
Infrarotled
3V3
3V3
D1
Photodiode
R1
1K
R2
10K
RASPI_RX
T1
BC548

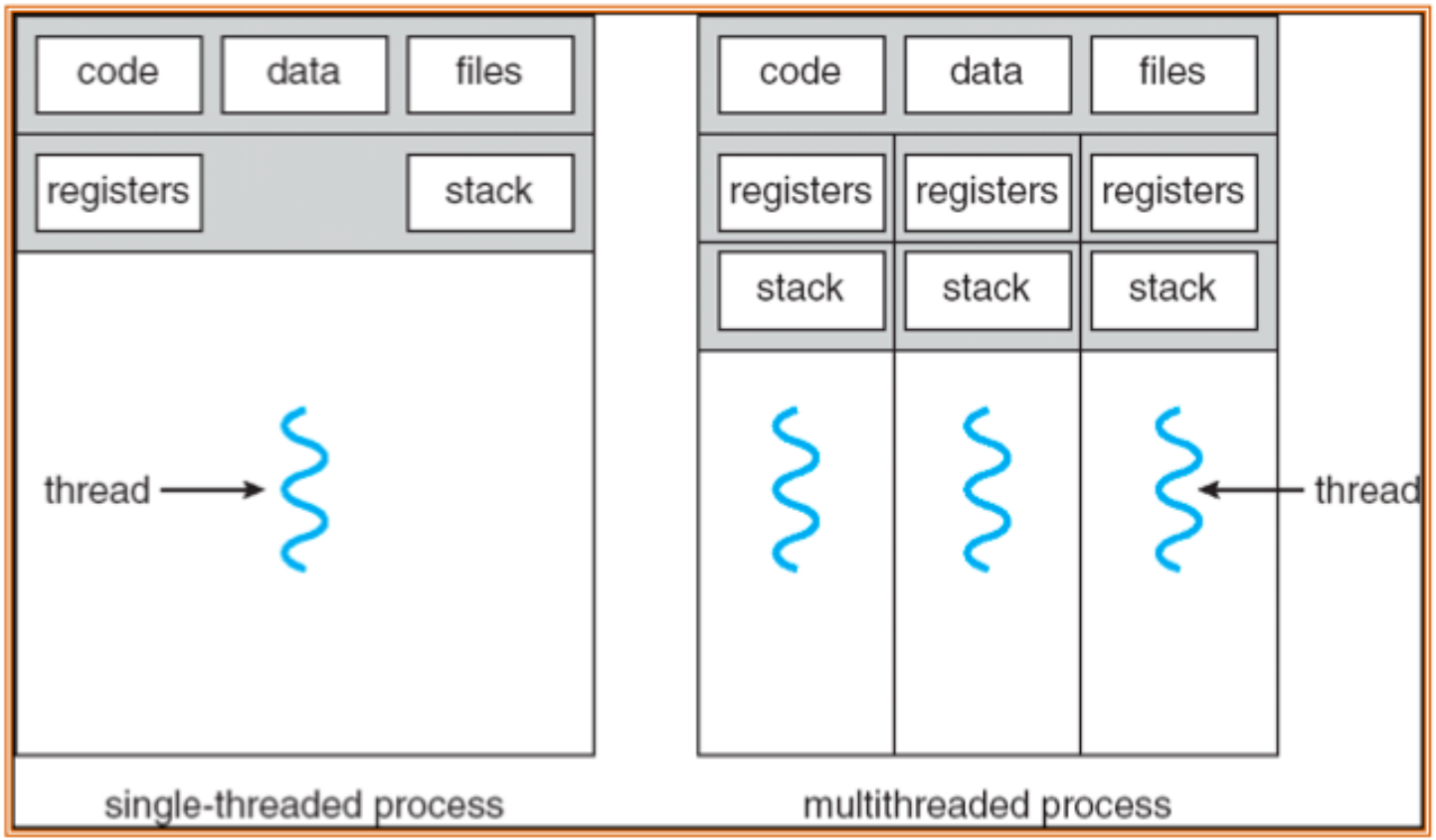
Hinweise zur Aufgabenstellung
zu Aufgabe 2

- Aufgabe 2 sollen Sie mit Hilfe von Threads lösen.

```
// Thread1
void *get_Data(void*){
    //Code zum Auslesen der seriellen Schnittstelle
}

//Thread2
void *send_Data(void*){
    //Code zum Senden von Daten über die serielle Schnittstelle
}
```

Hinweise zur Aufgabenstellung
Threads

- ein Thread (Faden) ist ein **Ausführungsstrang** und ist Teil eines Prozesses



A. Silberschatz et al., Operating Systems Concepts

Hinweise zur Aufgabenstellung
Threads

- Threads ermöglichen:
  - eine Aufteilung eines Prozesses in kleinere funktionale Einheiten
  - nebenläufiges oder paralleles abarbeiten von Aktivitäten innerhalb eines Prozesses
  - Schnelleres Umschalten führt zu besserem Antwortverhalten von Programmen
  - Einfache Benutzung gemeinsamer Ressourcen

Hinweise zur Aufgabenstellung
WiringPi und pThreads

- WiringPi stellt zwei Funktionen zur Verfügung um ein Thread zu erstellen/starten
  - PI_THREAD (myThread){

    // Code der auss

    }

```
#define  PI_THREAD(X)   void *X (void *dummy)
```

  - thread_id = piThreadCreate(my

    //Aufruf im Programm, Startet myThre

    // Returncode: 0 alles i.O.

    <0 Fehlerfall

```
int piThreadCreate (void *(*fn)(void *))
{
    pthread_t myThread ;
    return pthread_create (&myThread,
                           NULL, fn, NULL) ;
}
```

- mit pthread_join(myThread,NULL) kann auf das Ende des Threads gewartet werden

Quellen

- http://de.wikipedia.org/wiki/Stromz%C3%A4hler
- http://de.wikipedia.org/wiki/Ferraris-Z%C3%A4hler
- http://de.wikipedia.org/wiki/Intelligenter_Z%C3%A4hler
- http://dejure.org/gesetze/EnWG/21d.html
- http://www.easymeter.com/fileadmin/bilder/downloads/100125_Q3D_Produktblatt.pdf
- http://www.easymeter.com/haushaltszaehler.html
- http://de.wikipedia.org/wiki/UART
- http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter
- http://www.electronics-tutorials.ws/sequential/seq15a.gif
- http://www.hsg-kl.de/faecher/inf/netze/material/bitfuerbit.html
- http://openbook.galileocomputing.de/c_von_a_bis_z/026_c_paralleles_rechnen_004.htm
- http://www.mikrocontroller.net/attachment/89888/Q3Dx_D0_Spezifikation_v11.pdf
- http://www.edi-energy.de/files2%5COBIS-Kennzahlen-System%202.2a_20130401.pdf
- http://linux.die.net/man/3/termios
- http://openbook.galileocomputing.de/linux_unix_programmierung/Kap13-000.htm