Informatikwerkstatt, Foliensatz 4 Logikanalysator, Bluetooth, LC-Display und Sonar-Abstandssensor

G. Kemnitz

Institut für Informatik, TU Clausthal (IW4)

9. November 2016

Inhalt:

Interaktive Übungen:

1. Test mit dem Logikanalysator (USBLOGI)
2. Bluetooth (echo_bt)
3. Test des LC-Displays (test_lcd)
4. Test des Ultraschallsensors (test_sonar)

Wiederholung

Wiederholungsaufgabe 4.1: USART-Inititial.

```
void com_pc_init(){
 //Initialisierung USART 2 (PC-Kabelverbindung)
 UCSR2C=0b110;              //Übertragungsformat 8N1
 UBRR2=51;                  //9600 Baud
                            //Empf. und Sender ein
 UCSR2B=(1<<RXEN2)|(1<<TXEN2);
}
```

- Was bedeutet Übertragungsformat 8N1?
- Was ist die Baudrate?
- Wie lange dauert mindestens die Übertragung eines Bytes?

Für die nächste Folie:

```
#define symbol Rest der Zeile
```

Vor der Übersetzung wird »symbol« durch »Rest der Zeile« ersetzt.

Wiederholungsaufgabe 4.2: Empfang

1. Wozu dient 7 wie funktioniert die nachfolgende Funktion?

```
#define NEW_DAT (UCSR2A & (1<<RXC2))
uint8_t getByte(){
  while (!NEW_DAT);  //warte auf ein Byte
  return UDR2;       //Byte zurueckgeben
}
```

2. Vervollständigen Sie die nachfolgende Funktion zum Empfang einer vorzeichenfreien 2-Byte-Zahl.

```
uint16_t get2Byte(){
 uint16_t a;        //lokale Variable
 while (!NEW_DAT);//warte auf ein Byte


 return a;          //2-Byte-Wert zurueckgeben
}
```

Lösung

1. Funktion zum Empfang eines Bytes. Sie wartet, solange kein Byte im Empfangspuffer ist und gibt danach das Byte zurück.
2. Vervollständigtes Programm:

```
uint16_t get2Byte(){
 uint16_t a;          //lokale Variable
 while (!NEW_DAT);//warte auf ein Byte
 a = UDR2;            //als Byte 1 übernehmen
 while (!NEW_DAT);//warte auf ein Byte
 a = (a<<8)|UDR2; //als Byte 0 übernehmen
 return a;            //2-Byte-Wert zurueckgeben
}
```

Wiederholungsaufgabe 4.3: Sendefunktionen

1. Wozu dient und wie funktioniert die nachfolgende Funktion?

```
#define SB_FREE (UCSR2A & (1<<UDRE2))
void sendByte(uint8_t dat){
 while (!SB_FREE); //warte Puffer frei
 UDR2 = dat;       //Byte versenden
}
```

2. Vervollständigen Sie die nachfolgende Funktion zum versenden vorzeichenfreier 2-Byte-Zahlen, höherwertiges Byte zuerst:

```
void send2Byte(uint16_t dat){
 while (!SB_FREE); //warte bis Puffer frei




}
```

Lösung

1. Funktion zum Versenden eines Bytes. Sie wartet, solange der Sendepuffer voll ist und kopiert danach das Byte in den Sendepuffer.
2. Vervollständigtes Programm:

```
void send2Byte(uint16_t dat){
 while (!SB_FREE); //warte bis Puffer frei
 UDR2 = dat>>8;     //Byte 1 => Sendepuffer
 while (!SB_FREE); //warte bis Puffer frei
 UDR2 = dat & 0xFF;//Byte 0 => Sendepuffer
}
```

Wiederholungsaufgabe 4.4: Sende Zählstand

Die Funktion soll über USART2 beginnend mit null bei jedem Aufruf den um eins erhöhten Bytewert senden (Gedächtnis):

```
#define SB_FREE (UCSR2A & (1<<UDRE2))



void sendZaehlstand(){









}
```

Lösung

Funktion, die über USART2 beginnend mit null bei jedem Aufruf den um eins erhöhten Bytewert sendet:

```
#define SB_FREE (UCSR2A & (1<<UDRE2))

uint8_t zaehler=0; //Zustandsvariable

void sendZaehlstand(){
 while (!SB_FREE); //warte bis Puffer frei
 UDR2 = zaehler;   //Zählstand => Sendepuffer
 zaehler++;        //Zähler erhöhen
}
```

Wiederholungsaufgabe 4.5: Re-Engineering

1. Wie funktioniert das nachfolgende Programm?

```
int main(){
 com_pc_init();
 uint8_t i,dat, text[]="Ausgabetext\n";
 while(1){
  dat = getByte();
  for (i=0; i<(dat & 0b111); i++){
   sendByte(text[i]);
  }
 }
}
```

2. Welche Zeichenfolgen empfängt der PC, wenn er im Abstand von 1 Sekunde die Zeichen 'A', 'F', und 'J' sendet?

Lösung

1. Funktion des Programms:
   - Warten auf ein Byte vom PC.
   - Zählschleife von 0 bis zum Zahlenwert der niederwertigsten 3 Bits des Bytes (z).
   - In jedem Schleifendurchlauf werden die ersten z Zeichen der Zeichenkette "Ausgabetext" versendet.
2. Programmausgabe:

empf. Zeichen	z	Zeit	gesendete Zeichenfolge
'A'(0x41)	1	0	"A"
'F'(0x44)	6	1 s	"Ausgab"
'J'(0x4A)	2	2 s	"Au"

Test mit Logikanalysator

Untersuchung von Zeitabläufen mit Logikanalysator

Genauere Zeituntersuchungen, z.B.

- Zeitmessungen von der Ausgabeanweisung bis zum seriellen Versenden,
- Kontrolle des Protokolls und der Baudrate, ...

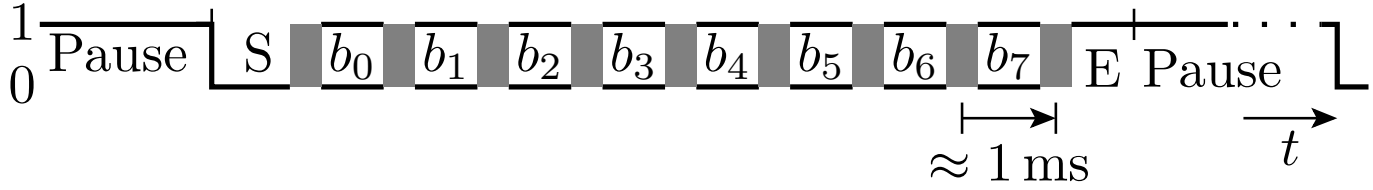
erfordern einen Logikanalysator. Ein Logikanalysator zeichnet binäre Signalverläufe auf.

Der hier verwendete USB-LOGI-500

- zeichnet bis zu 32 binäre Signale mit maximal 600 Millionen Abtastwerten pro Sekunde auf.
- PC-Verbindung über USB.
- Konfiguration über ein XML-Script.
- Ansteuerung auf der Komandozeile. ...

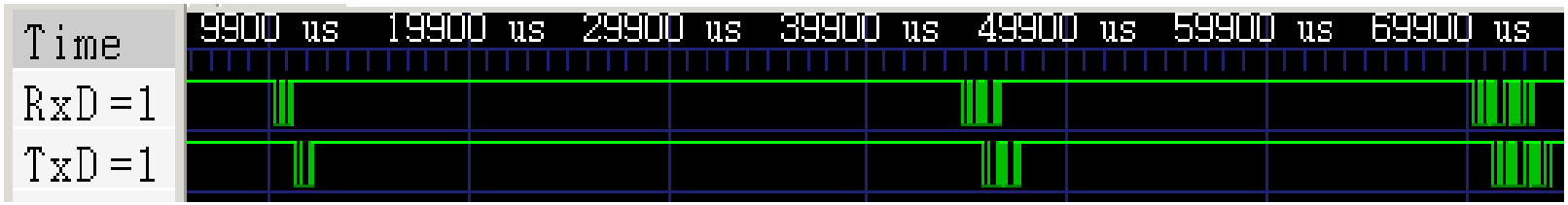
Arbeit mit einem Logikanalysator

- Abschätzung des Signalverlaufs, z.B. USART 8N1, 9600 Baud:



- Anschluss von LA-Eingängen an alle zu beobachtenden Signale. Aktivierung dieser LA-Eingänge im Konfigurationsscript und Zuordnung von Signalnamen.
- Wahl der Abtastrate so, dass zwischen zwei aufeinanderfolgenden Signalwechseln mehrfach abgetastet und der komplette interessierende Signalbereich aufgezeichnet wird.
- Wahl einer Trigger-Bedingung, die nach LA-Aufzeichnungsstart erstmalig im aufzuzeichnenden Zeitfenster erfüllt ist.
- Pre-Trigger: Festlegung des Aufzeichnunganteils vor dem Trigger-Ereignis, Wertebereich 0, $\frac{1}{8}$, ... 1, Standardwert $\frac{1}{8}$.

RxD und TxD bei Abarbeitung von »scom_txy.py«



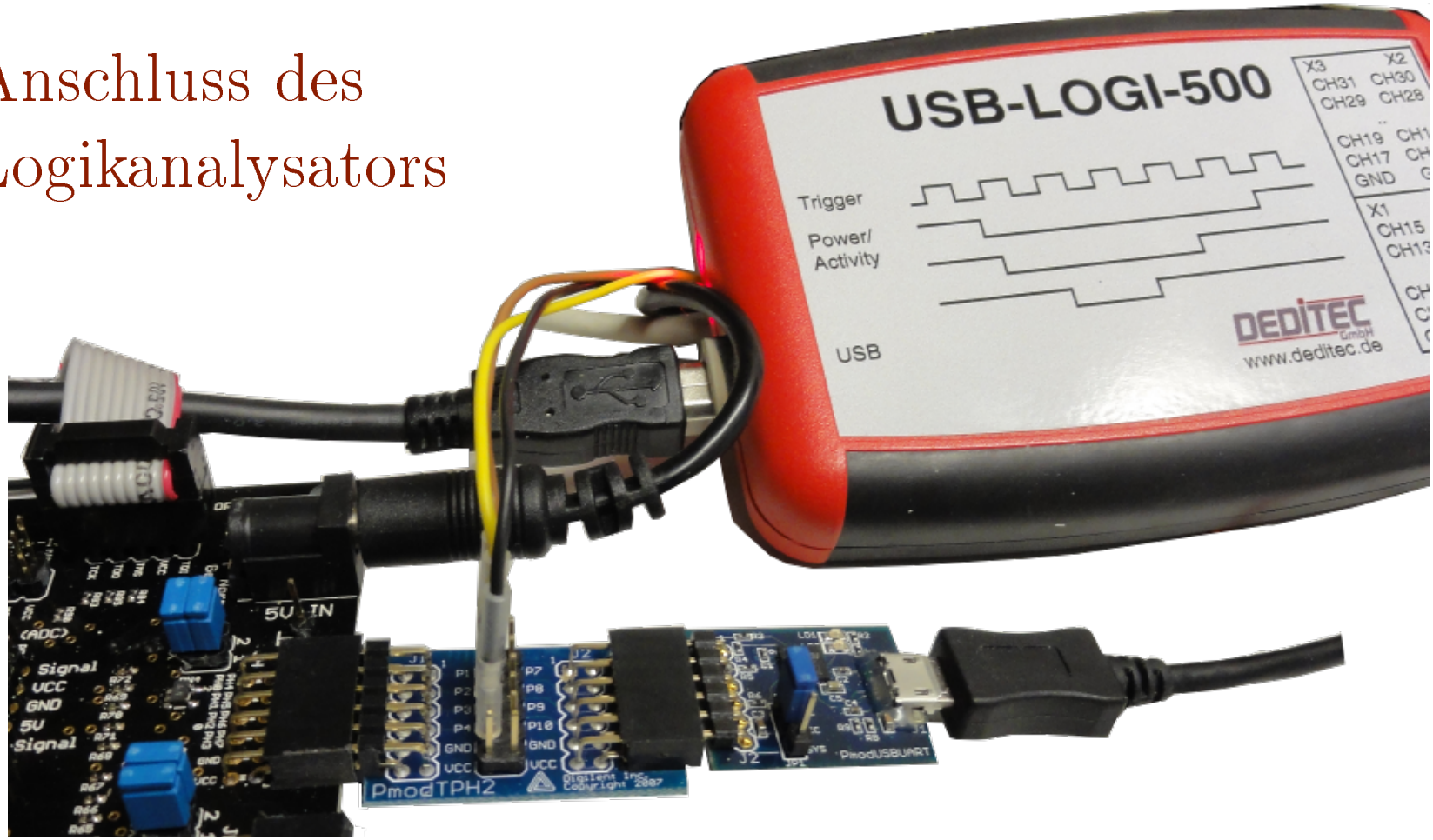
RxD – serielle Daten vom FDDI-Chip zum Mikrorechner; TxD – zum FDDI-Chip zurückgesendete Echo-Daten.

- Aufzeichnungstakt: 50000 Werte pro Sekunde.
- Trigger: erste fallende Flanke von RxD. Eindeutig, wenn zuerst der LA und dann das Python-Programm gestartet wird.
- Pre-Trigger: 1/8 (Standardwert, Angabe nicht erforderlich).

Beobachtungen:

- Bytes eines Pakets werden lückenfrei gesendet, dazwischen längere Pausen.
- Echo praktisch sofort nach Byte-Empfang

Anschluss des Logikanalysators



PModUSBUSART über Zwischenadapter PmodTPH2, daran

- GND (schwarz) an Masse (Adapter Gnd),
- CH0 (gelb) an RxD (Adapter P3) und
- CH1 (braun) an TxD (Adapter P2) anstecken.

Konfiguration des USB-Logi in »echotest.xml«

- Aufzeichnungstakt: 50000 Werte pro Sekunde.
- CH0: RxD, CH1: TxD
- Trigger: erste fallende Flanke von RxD.
- Pre-Trigger: 1/8 (Standardwert).

```
<la>
 <samplerate>50000</samplerate>
 <signals>
  <signal name="RxD"><ch>0</ch></signal> #RxD => CH0
  <signal name="TxD"><ch>1</ch></signal> #TxD => CH1
 </signals>
 <trigger when="A">
  <A> <ch when="falling_edge">0</ch> </A>
 </trigger>
</la>
```

(Doku für Konfig-Scripte siehe WEB-Seite).

Messung durchführen

- Auf dem Mikrorechner das Echo-Programm starten.
- Eine Windows-Konsole (cmd.exe) für Python und eine für den LA starten.
- In der Python-Konsole zum Verzeichnis »...\Informatikwerkstatt\Python« und in der LA-Konsole zum Verzeichnis »...\Informatikwerkstatt\P04\USBLOGI« wechseln. USB-Logi starten mit dem Kommando:

```
usb-logi echotest.xml
```

- Sobald der USB-Logi auf sein Triggerereignis wartet, in der Python-Konsole »scom_txy« starten.

2. Test mit Logikanalysator

- Start des Logikanalysators:

```
H:\Informatikwerkstatt\USBLOGI>usb-logi echotest.xml
+=========================================================+
| USB-LOGI-500 Logik Analysator                           |
+=========================================================+
        xml-Datei prüfen, Verbindung herstellen, ...
Fülle Pre-Trigger...

                                                        100 %

Warte auf Trigger...
```

- Start des Python-Scripts:

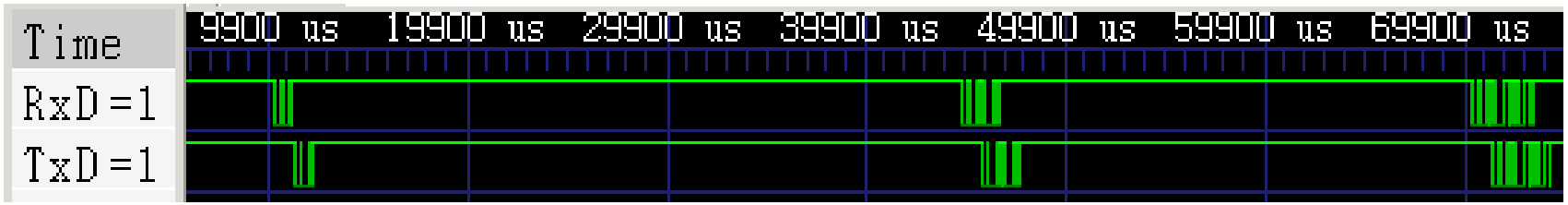
```
H:\Informatikwerkstatt\Progr_IW\Python>scom_txy.py
Empfangene Daten: D dt= 13.2549278489 ms
Empfangene Daten: Da dt= 14.921187855 ms
Empfangene Daten: Das dt= 14.4043923316 ms
Empfangene Daten: Das  dt= 14.4265738178 ms
                  . . .
```

- Bei der ersten fallenden Flanke »RxD« beginnt die Aufzeichnung.

2. Test mit Logikanalysator

```
Warte auf Trigger...
Aufzeichnung läuft...
                                                  100 %
+-----------------------------------------------------+
| (4) Erzeuge Waveform Dateien...                     |
+-----------------------------------------------------+
LXT Datei ./echotest.lxt erzeugt.
Bestehende SAV Datei ./echotest.sav beibehalten.
+-----------------------------------------------------+
| (5) Starte GTKWave...                               |
+-----------------------------------------------------+
```

- Entspechend Einstellung werden 1/8 der Werte vor und 7/8 der Werte nach dem Trigger-Ereignis aufgezeichnet. Im sich öffnenden GTKWave-Fenster mit Zoom und Scroll-Leisten Darstellung anpassen.

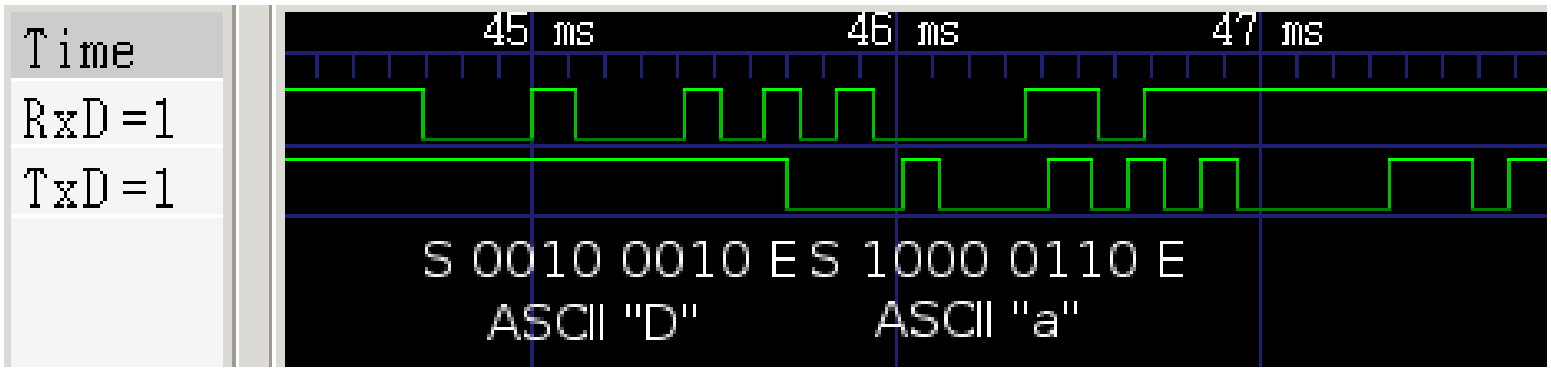


1 Byte nach ca. 10 ms (Pre-Trigger-Zeit), 2 Bytes nach ca. 50 ms, ...

2. Test mit Logikanalysator

- 1-Byte-Paket nach ca. 10 ms ($\frac{1}{8}$ der Aufzeichnungsdauer),
- 2 Byte-Paket nach ca. 50 ms und
- 3-Byte-Paket nach ca. 75 ms.

Mit Zoom vergrößertes 2-Byte-Packet:



Übertragene Zeichenfolge »Da«. Mikrorechner beginnt mit Rückübertragung des ersten Bytes nach der halben Stoppbitzeit.

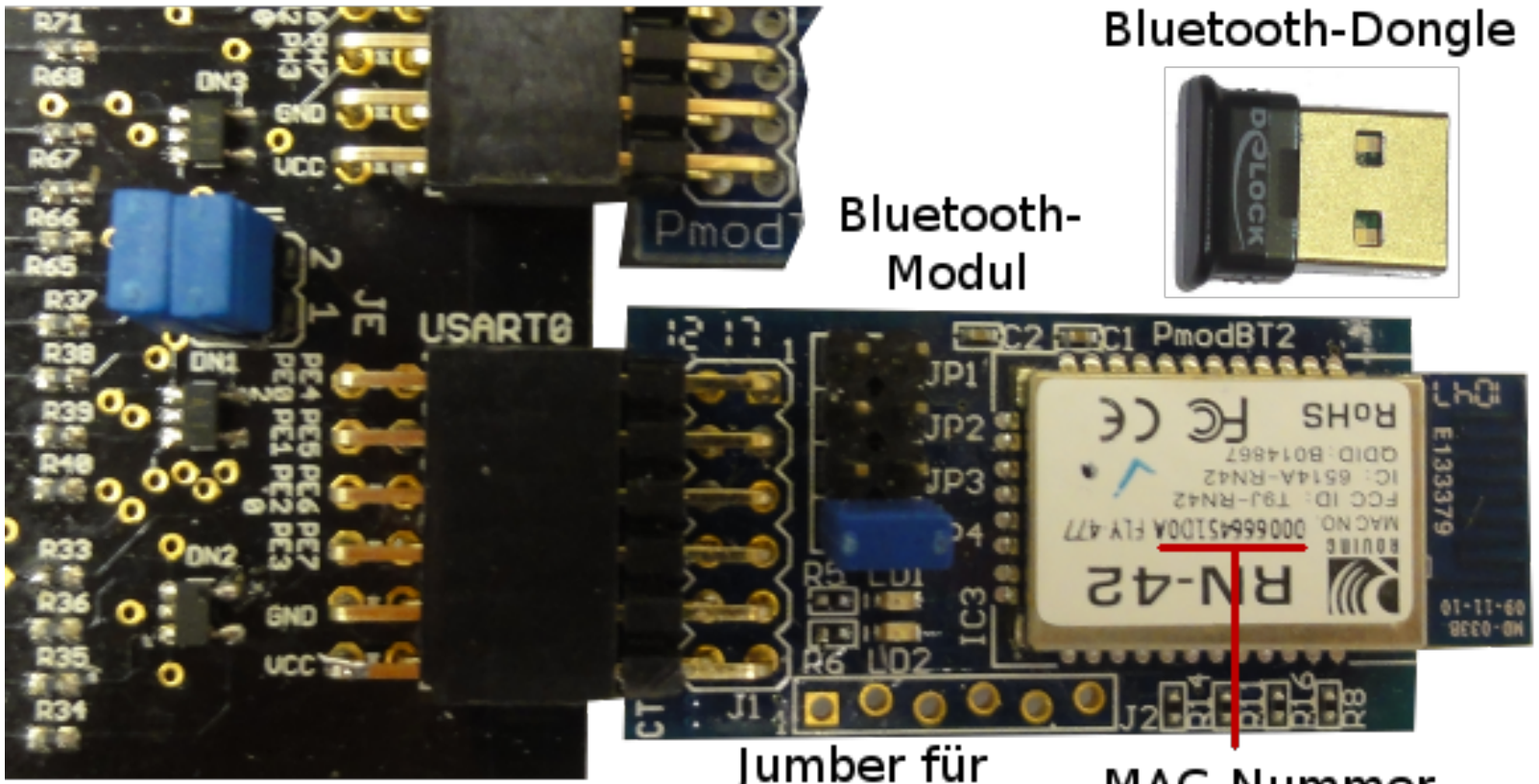
Bluetooth

Bluetooth

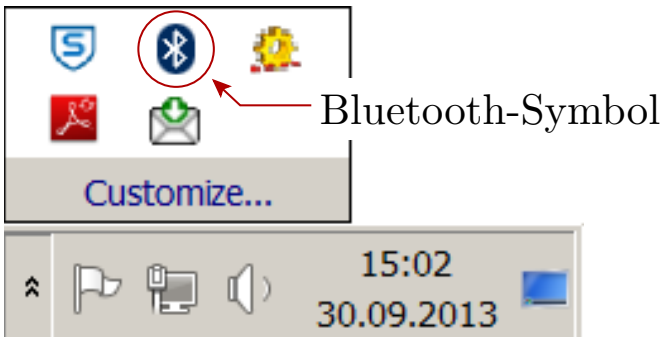
- Bei Bluetooth wird die serielle Übertragung anstatt über eine USB- über eine Funkverbindung getunnelt.
- Statt an USART2 (Stecker JH, Port H) wird wird das Bluetooth-Modul auch in allen weiteren Projekten an USART0 (Stecker JE, Port E) gesteckt.
- Kabel und Funkverbindung können so bei Bedarf gleichzeitig genutzt werden.
- In den Kommunikationsprogrammen ist für die Umstellung auf Bluetooth jeweils USART2 durch USART0 zu ersetzen.

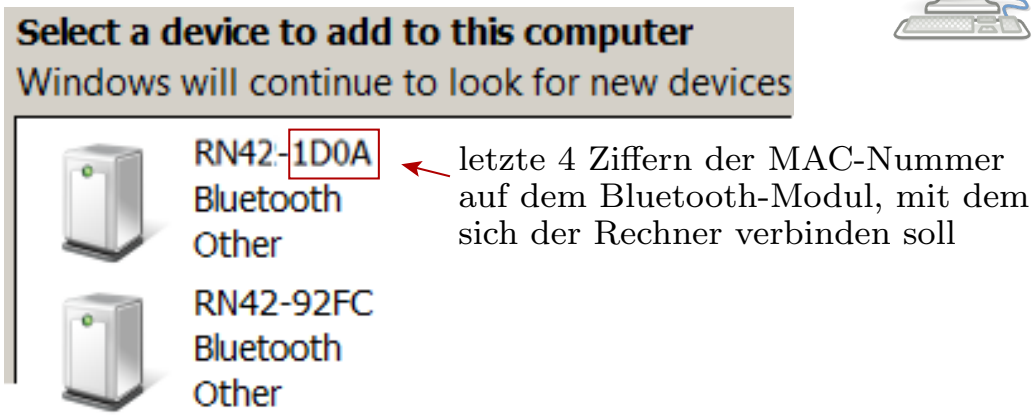
Bluetooth-Modul anschließen

- PmodBT2 an JE (USART0) stecken und Jumper JEX »gekreuzt (=)«.
- Bluetooth-Dongle in den PC stecken.



Bluetooth-Verbindung auf PC einrichten[2]

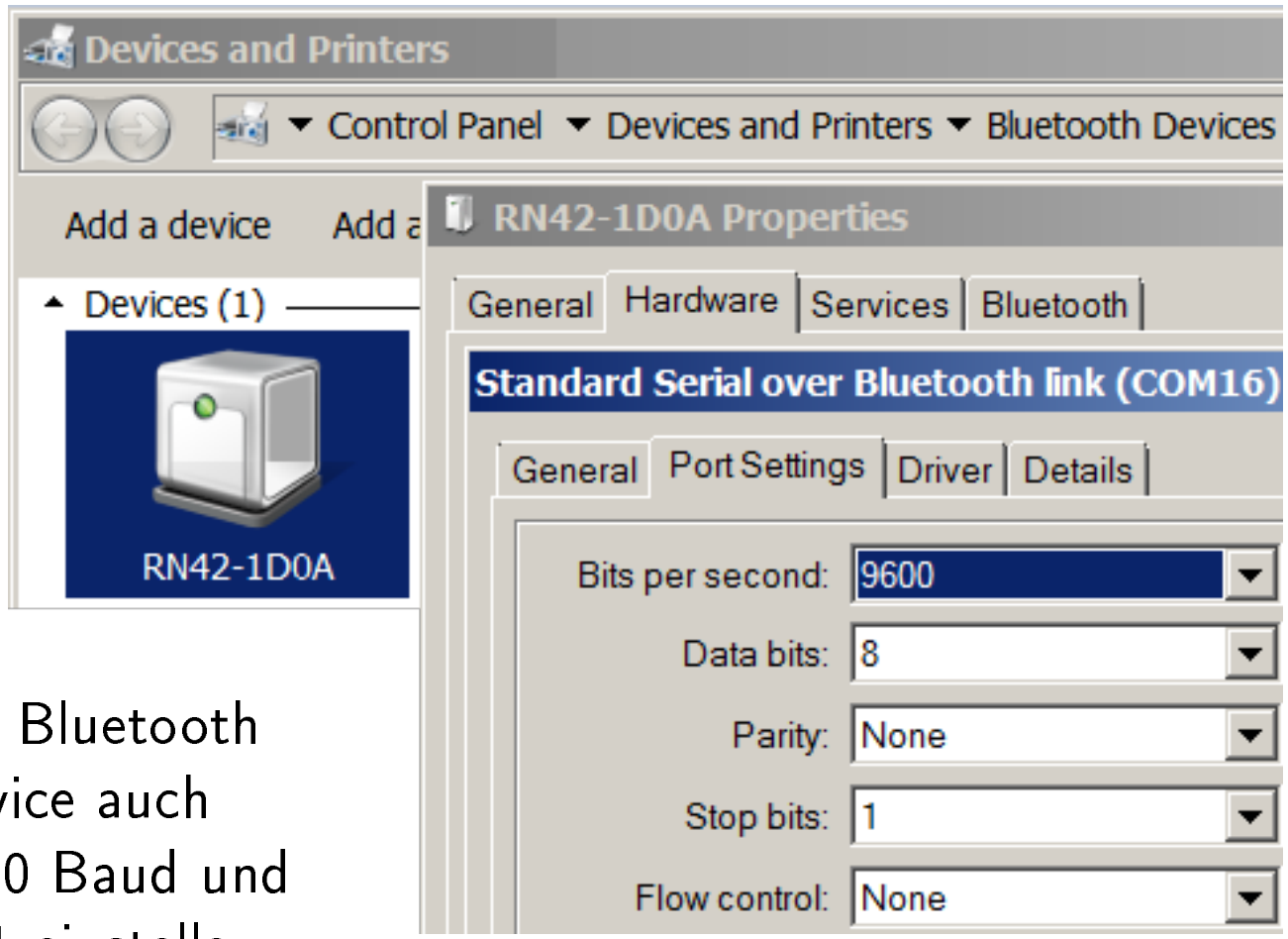




- Unter Windows Doppelklick auf Bluetooth-Symbol.
- Show Bluetooth Devices (Bei ersten mal »Add a Device«).
- Device mit der MAC-Nummer auf dem PMOD auswählen.
- Für »Enter the Device Pairing Code« Eingabe »1234«.[1]
- recher Mouse-Click > Properties > Hardware; hinter dem Namen COM-Port ablesen.
- HTerm: abgelesener COM-Port, 9600 Baud, 8N1, »Connect«.

[1] Fehler, wenn mit der Eingabe zu lange gewartet wird.

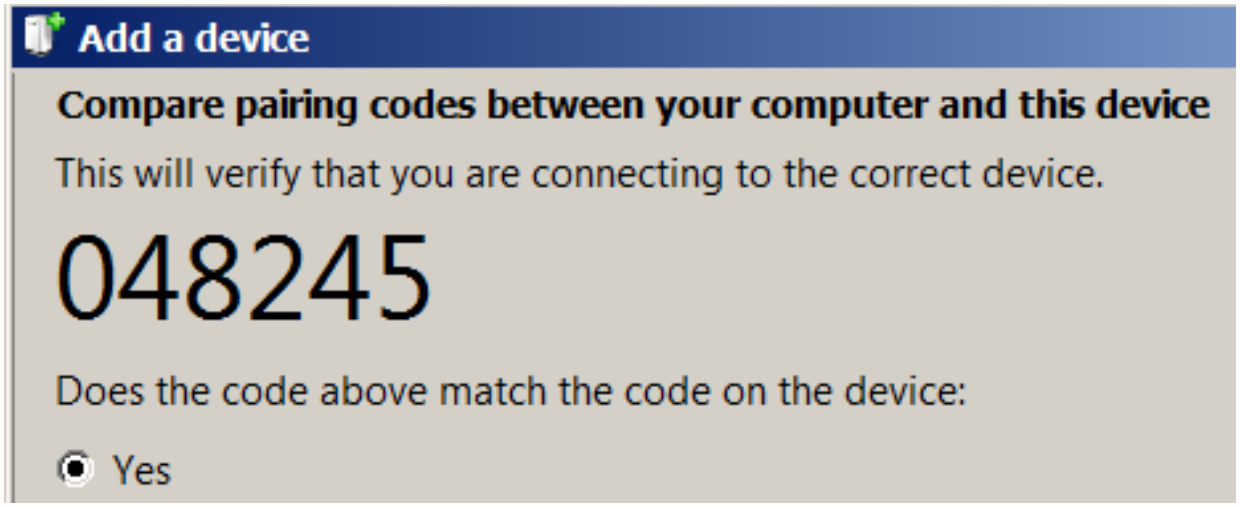
[2] Gilt nur für die schon vor längerer Zeit gekauften Module. Bei den neueren ist der Pairing Codenur zu bestätigen (siehe später Folie 28).



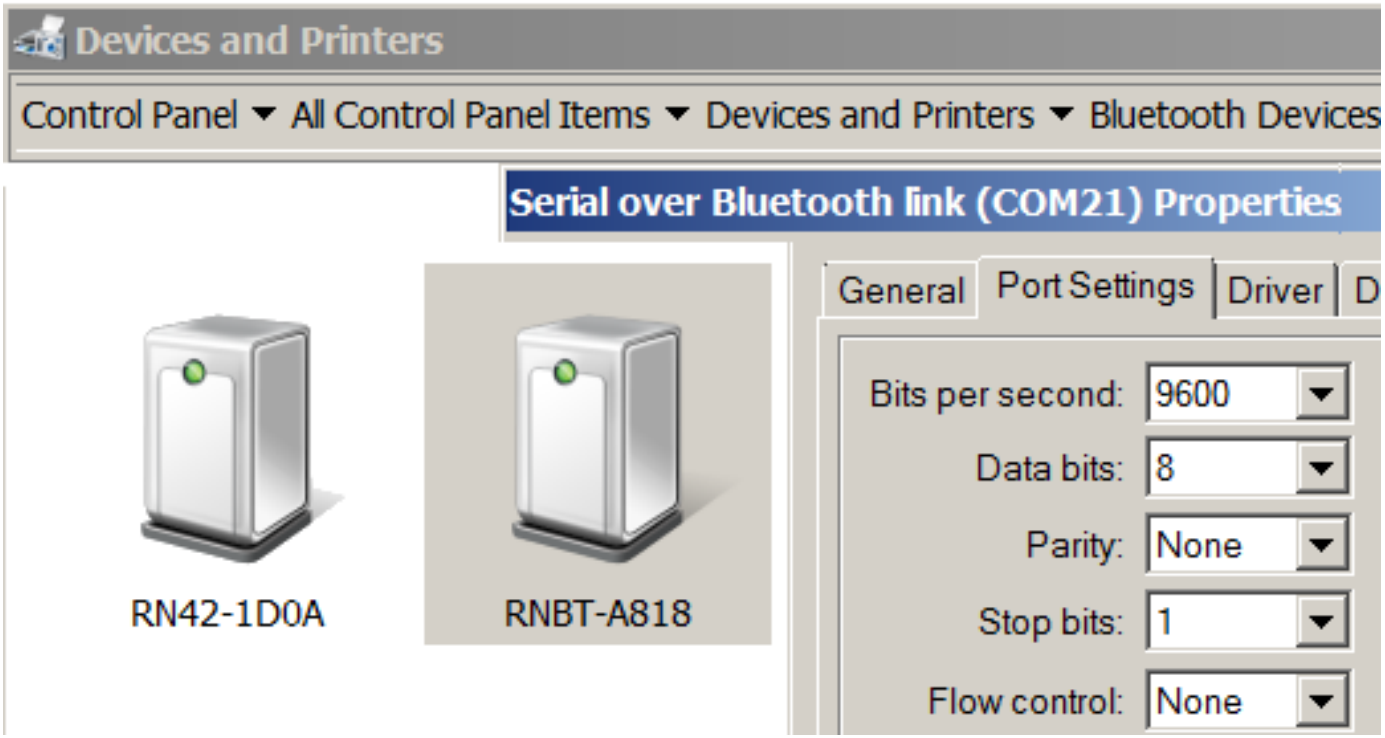
- Für Bluetooth Device auch 9600 Baud und 8N1 einstellen.
- In ATMEL-Studio Programm »echo_bt« starten[3].
- Mit HTerm Zeichen senden. Empfang + LED-Ausgabe der Zeichenanzahl kontrollieren.

[3] Bisheriges Echoprogramm für USART0 statt USART2.

Bluetooth-Verbindung mit neueren Modulen



- Nach »Add a device« ist der »Pairing Code« nur zu bestätigen.
- Im Menü »Devices and Printer« Rechtsklick > Properties >Hardware: Com-Port ablesen
- weiter unter Proporties > PortSettings: Protokolleinstellung: 8N1, 9600 Bd kontrollieren / korrigieren.



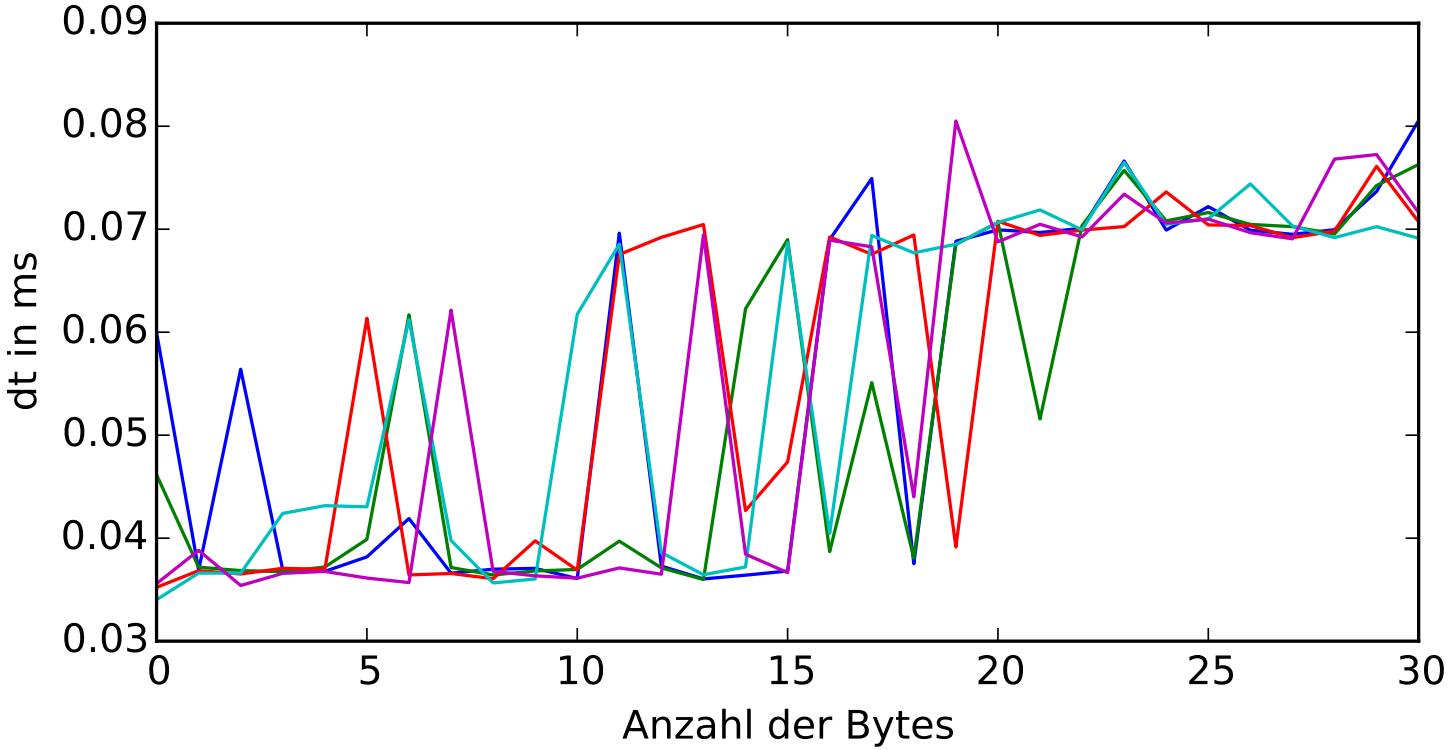
Wenn sich der Port im HTerm nicht öffnen lässt, warten, bis der Treiber fertig installiert ist.

Untersuchung der Übertragungsdauer

- Programm »echo_bt.c« weiter laufen lassen.
- Programm »Python\scom_txy5.py« im Editor öffnen. COM-Port durch den für die Bluetooth-Verbindung ersetzen.
- HTerm »Disconnect«.
- Windows-Konsole (cmd) starten. In das Verzeichnis ...\Python wechseln. Programm »scom_txy5.py« starten:

```
H:\Informatikwerkstatt\Python>scom_txy5.py
Empfangene Daten: D dt= 62.1397588383 ms
Empfangene Daten: Da dt= 60.5297850124 ms
Empfangene Daten: Das dt= 36.0593746958 ms
                  ...
```

- Graphische Darstellung der Zeitmesswerte nächste Folie.



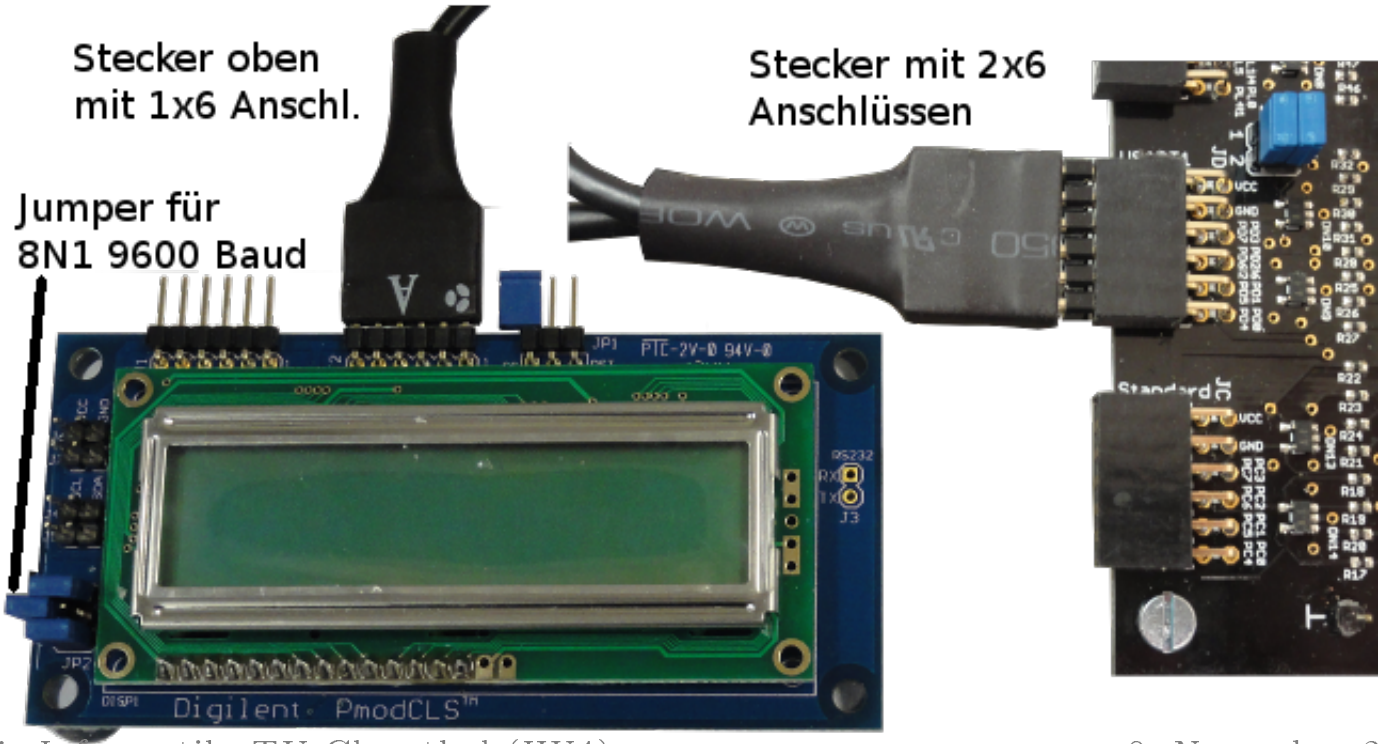
- Die Übertragungsdauer hat einen zufälligen Wert.
- Für 21 bis 28 Byte große Pakete etwa doppelte Dauer (≈70 ms) im Vergleich zur Tunnelung durch USB.
- Ändert sich die Übertragungsdauer, wenn mehrere Übungsgruppen zeitgleich Bluetooth nutzen?

LC-Display

LC-Display

Das LC-Display hat eine serielle Schnittstelle am Stecker J2, von der nur die Sendeeinheit benötigt wird:

- MR-Board JD oben (USART1) über Y-Kabel an LCD J2.
- Für USART-Protokoll 8N1, 9600 Baud auf LCD Jumper JP2 MD0 und MD2 stecken.



Testprogramm »test_lcd.c«


```
void lcd_init(){          //Initialisierung USART1
  UCSR1C = 0b110;         //Übertragungsformat 8N1
  UBRR1  = 51;            //9600 Baud
  UCSR1B = (1<<TXEN1); //Sender ein
}
uint8_t LCD_dat[]=    // Zeichenkette zur Ausgabe
  "\x1B[0h\x1B[j"     // 7-Zeichen-Init.-String
  "A=... Err:....."  // 1. Zeile (16 Zeichen)
  "S:... Inp:.....";// 2. Zeile (16 Zeichen)
... // main() siehe nächste Folie
```

"\x1B[0h" – Zeilenumbruch nach 16 Zeichen; "\x1B[j" – Anzeige löschen, Cursor auf erstes Zeichen; siehe WEB-Seite > Pmod...: PmodCLS_rm.pdf.

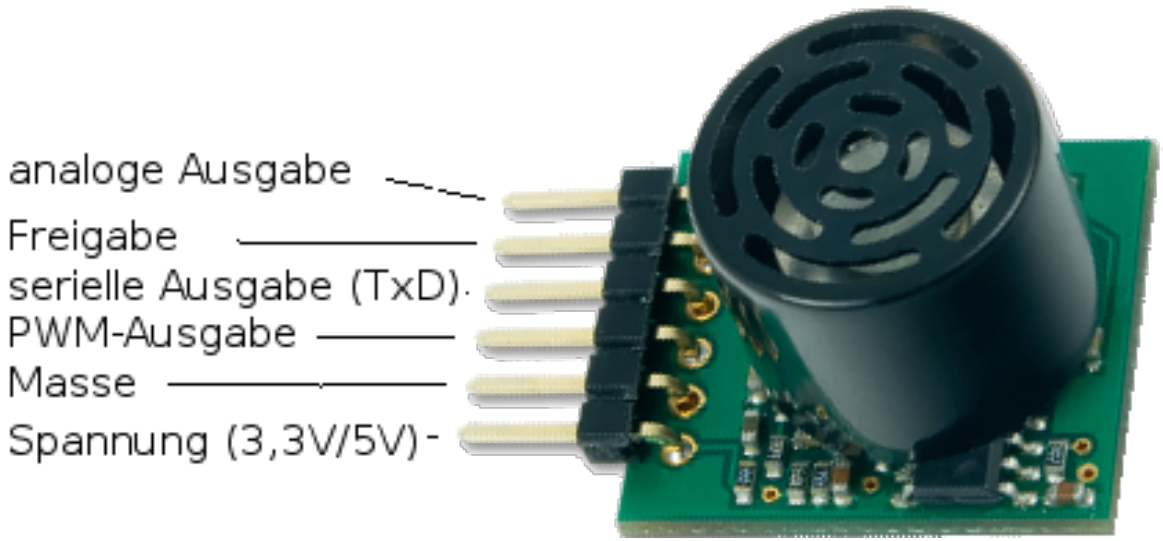
```
uint8_t idx;                          //Indexvariable
int main(void){
 lcd_init();                          //USART1 initial.
 for(idx=0;idx<39;idx++){             //für alle 39 Zeichen
  while(!(UCSR1A&(1<<UDRE1)));//warte Puffer frei
  UDR1 = LCD_dat[idx];                //Zeichen in Puffer
 }
}
```

Das LC-Display dient im Weiteren für die Anzeige von Fahrzeugsteuerzuständen, Fehlermeldungen, Sensorwerten, ...

- Projekt »F4-test_lcd\test_lcd« öffnen.
- Programm übersetzen und ausführen.
- LCD-Ausgabe kontrollieren.

Sonar-Sensor

Sensormodul PmodMAXSONAR



- PmodMAXSONAR über Y-Kabel an JD unten (USART1) anstecken. JDX auf »gekreuzt (=)«
- Wie auf der nächsten Folie gezeigt, Multimetereingang an Pin 1 und Multimetermasse (COM) an einen Masseanschluss (⊥) des Mikrorechner-Boards.
- Spannung zuschalten.

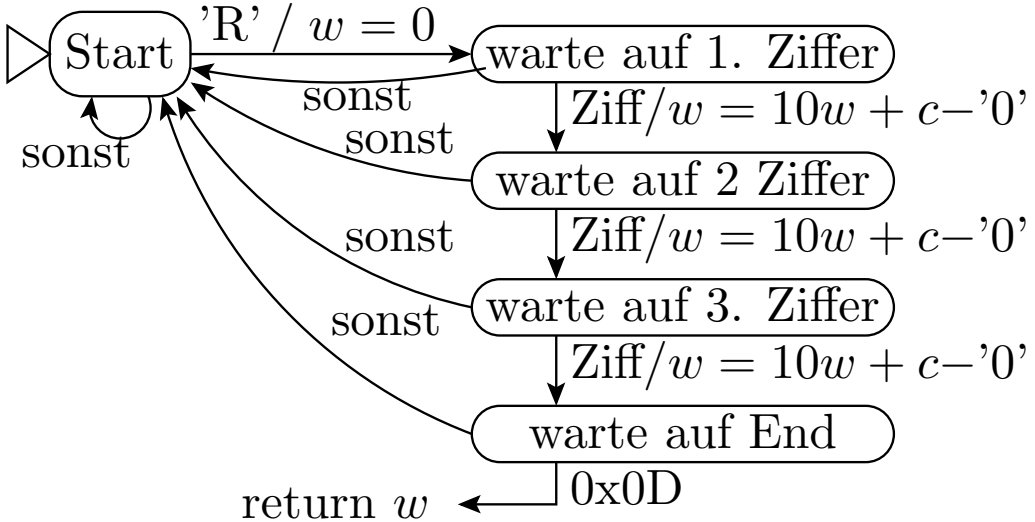
- Projekt »F4-test_sonar\test_sonar«, übersetzen, im Debugger starten und anhalten.
- Zur Aktivierung PD5 (Freigabesignal) im Debug-Modus als Ausgang und auf eins setzen.
- Hand im Anstand $>15\,\mathrm{cm}$ vor dem Sensor bewegen.

Spannung am Multimeter: $3{,}3\,\mathrm{V} \cdot x/512" \approx 2{,}5\,\frac{\mathrm{mV}}{\mathrm{cm}}$. Messbereich 6" bis 254" ($1" = 2{,}54\,\mathrm{cm}$). Mindestabstand $\approx 15\,\mathrm{cm}$. Auflösung 1" (" – Zoll, engl. Inch).

Der Mikrorechner empfängt an USART1 eine Zeichenfolge mit dem Format:

$$\text{'R'}, z, z, z, \text{'\textbackslash 13'}$$

('R' – ASCII-Zeichen; z – Ziffer 0..9 als ASCII-Zeichen; '\13' – Zeilenumbruch). Ablaufgraph für die Decodierung des Abstandswerts:



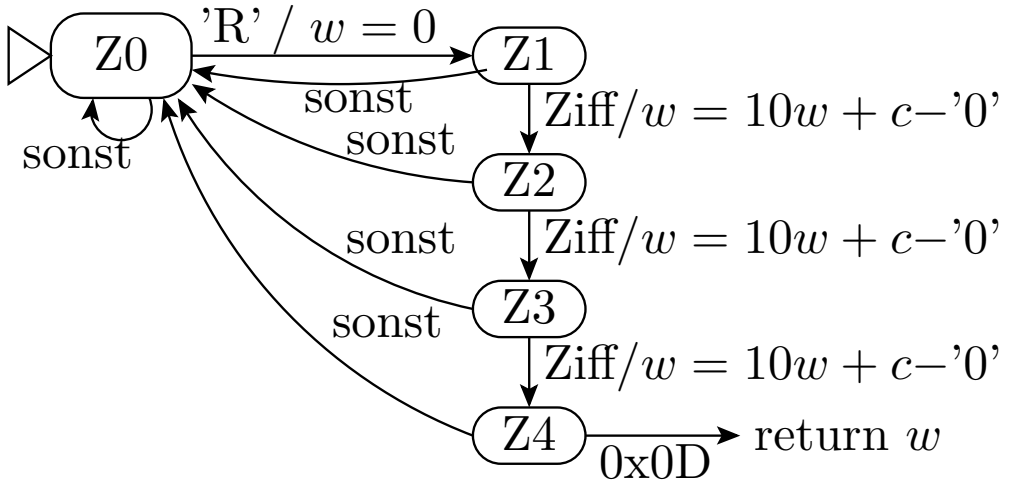
Übergangsbedingung letztes empfangenes Zeichen c hat den Wert ...

Ziff: $c \in \{\text{'1'},\text{'2'},\ldots\text{'9'}\}$

Ziffernwert ist Byte-Code minus Byte-Code von '0'

In w wird der Sensorwert berechnet.

Automaten



Ablaufgraphen werden als Automaten programmiert (Berechnung des Folgezustands und der Ausgabe aus dem Istzustand und der Eingabe). Im weiteren Studium werden Automaten behandelt:

- in der Technischen Informatik als Funktionsmodelle für Schaltungen mit Zustandsspeicher,
- in Software-Technik zur Spezifikation von Zielfunktionen und
- in der Theoretischen Informatik für Sprachen, Grammatiken, ...

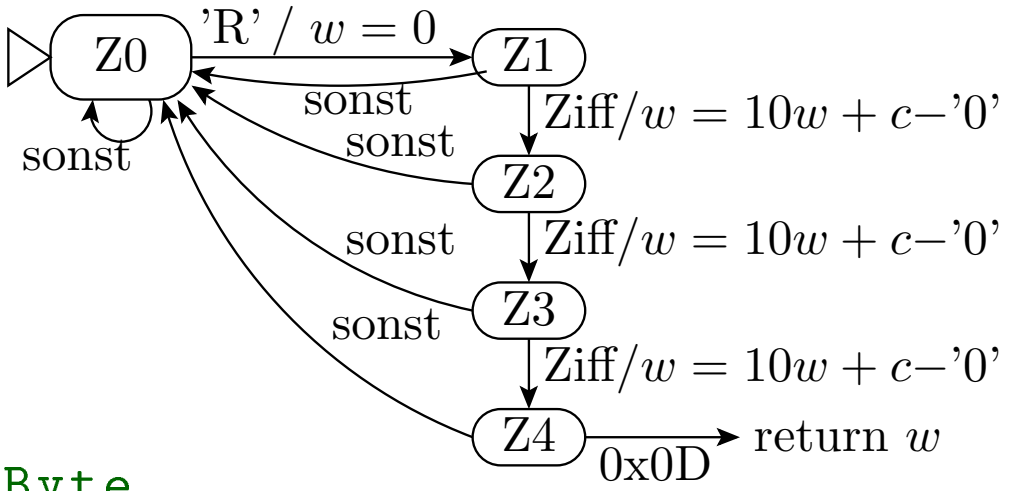
Testprogramm »test_sonar.c«

```
int main(){
  sonar_init();                 //Sensor initialisieren
  DDRJ = 0xFF;                  //LED-Ausgabe an Port J
  while(1){
    uint16_t s = getSonar(); //Wert abholen
    PORTJ = s & 0xFF;           //Wert ausgeben
  }
}
```

Initialierung des seriellen Empfangs und der Sensorausgabe:

```
void sonar_init(){
  UCSR1C=0b110;             // Übertragungsformat 8N1
  UBRR1=51;                 // 9600 Baud
  UCSR1B=(1<<RXEN1);        // Empfänger ein
  DDRD  |= 1<<PD5;          // PD5 Ausgang
  PORTD |= 1<<PD5;          // Sonar einschalten
}
```

- Empfangs-automat:




```
uint16_t getSonar(){
 uint8_t c, z=0;
 uint16_t w;
 while(1){
  while //Warte auf Byte
    (!(UCSR1A & (1<<RXC1)));
  c = UDR1;  //Byte Lesen
   if (z==0 && c=='R'){  //Z0 => Z1
    w=0; z=1;}
   else if                  // Z1 bis Z4 und Ziffer
     (z>0 && z<4 && c>='0' && c<='9'){
    z++; w=10*w+c-'0';}
   else if                  // Z5 und 0x13
    (z==4 && c==0x0D){
     return w;}
   else z=0;                // sonst-Kanten
 }
}
```

Aufgaben

Aufgabe 4.1: LA-Test Blockübertragung

- Schreiben Sie ein Mikrorechnerprogramm, das auf zwei Bytes vom PC wartet und nach Empfang beide Bytes in umgekehrter Reihenfolge zurückschickt.
- Testen Sie das Programm zuerst mit HTerm.
- Schließen Sie wie ab Folie 17 den USB-LOGI an (CH0: RxD, CH1: TxD) und bestimmen Sie mit der Konfigurationsdatei »echotest.xml« die Signalverläufe von RxD und TxD.
- Zeichnen Sie die Signalverläufe beim Test mit HTerm auf.

Aufgabe 4.2: Fortsetzung LA-Test

Erweitern Sie das Programm aus der Aufgabe zuvor um Testausgaben zur Ablaufkontrolle mit dem Logikanalysator auf Port G:

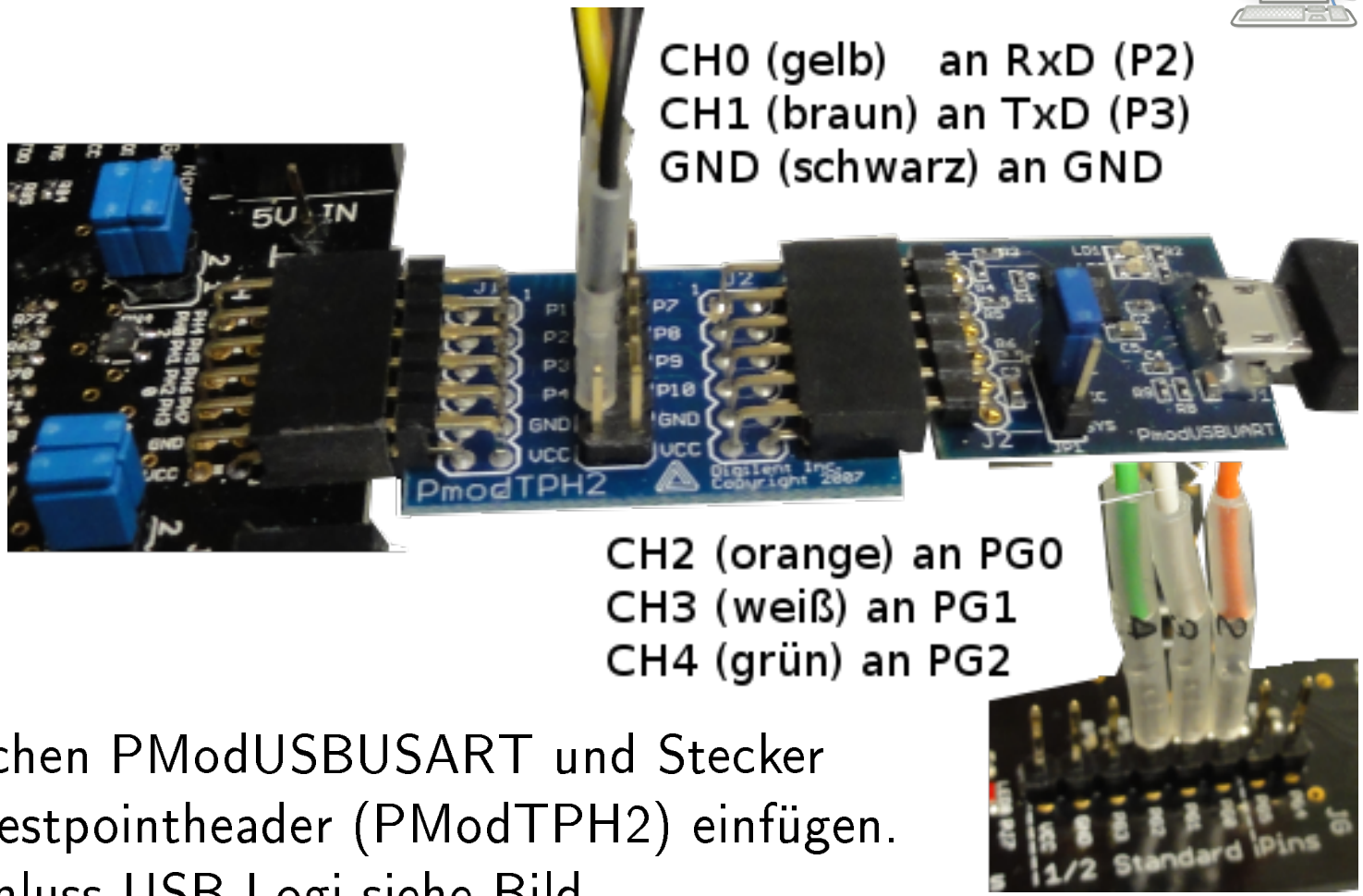
```
<erforderliche Header, und Variablenvereinb.>
int main(){
 DDRG = 0xFF; PORTG= 0; //Port G als Ausgang
 <USART2 initialisieren>
 while(1){                    //Zustandsausgabe
  <1. Byte empfangen>;   PORTG = 1;
  <2. Byte empfangen>;   PORTG = 2;
  <2. Byte senden>;      PORTG = 3;
  <1. Byte senden>;      PORTG = 4;
 }
}
```

Zeichnen Sie mit dem LA zusätzlich zu RxD und TxD die Zustandsausgabe an PG0 bis PG2 auf.

LA-Konfiguration und Anschluss (echo2b.xml)

```
<la>                                  # P04/USBLOGI/echo2b.xml
 <samplerate>500000</samplerate># 500.000 Werte/s
  <pretrigger>6</pretrigger>      # 6/8 Vortrigger-
  <signals>                       # aufzeichnung
    <signal name="Zustand">       # 3-Bit-Bus
      <ch>2</ch>                  # orange
      <ch>3</ch>                  # weiss
      <ch>4</ch>                  # grün
    </signal>
    <signal name="RxD"> <ch>0</ch> </signal> #gelb
    <signal name="TxD"> <ch>1</ch> </signal> #braun
  </signals>
 <trigger when="A">
  <A> <ch when="falling_edge">0</ch> </A>
 </trigger>
</la>
```

Anschluss des Logikanalysators



- Zwischen PModUSBUSART und Stecker JH Testpointheader (PModTPH2) einfügen.
- Anschluss USB-Logi siehe Bild.
- USB-Logi muss, wenn angesteckt, auch am USB-Kabel stecken. Sonst werden kontaktierte Signale auf null gezogen.

LA-Aufzeichnung durchführen

- Auf dem Mikrorechner das Echo-Programm starten.
- HTerm starten, Verbindung herstellen, zum ausprobieren 2 Bytes senden und kontrollieren, dass diese in umgekehrter Reihenfolge empfangen werden.
- Eine Windows-Konsole (cmd.exe) für den LA starten, zum Verzeichnis »...\Informatikwerkstatt\P04\USBLOGI« wechseln. USB-Logi starten mit dem Kommando:

```
usb-logi echo2b.xml
```

- Sobald der USB-Logi auf sein Triggerereignis wartet, die 2 Bytes nochmal senden.

Aufgabe 4.3: Bluetooth

- Stecken Sie das Bluetooth-Modul wie auf Folie 25 nur zusätzlich über das dem PmodTPH2, an dem der LA steckt, an JE (USART0) und den USB-Bluetooth-Dongle in den PC.
- Stellen Sie in der beschriebenen Weise eine Bluetooth-Verbindung her.
- Ändern Sie in dem Programm aus Aufgabe 4.2 überall USART2 in USART0 und wiederholen Sie die Tests mit dem COM-Port der Bluetooth-Verbindung.
- Zeichnen Sie auch hier wie in Aufgabe 4.2 die Signalverläufe von TxD, RxD und den kontaktiereten Anschlüssen von Port G auf.

Aufgabe 4.4: Hallo LC-Display

Schreiben Sie ein Programm, das auf dem LC-Display den Text »Hallo LC-Display« ausgibt.

Aufgabe 4.5: LC-Monitor

Schreiben Sie ein Programm, das auf Zeichen von der seriellen Bluetooth-Schnittstelle wartet und diese auf das LC-Display schreibt.

- Bei mehr als 16 Zeichen sollen die letzten 16 Zeichen der Zeile angezeigt werden.
- Bei ersten <Enter> soll auf Zeile 2 weitergeschrieben werden.
- Bei jedem weiteren <Enter> soll die untere Zeile nach oben kopiert, danach die untere Zeile gelöscht und ab Position 1 neu beschrieben werden.
- Testen Sie das Programm mit HTerm.

Aufgabe 4.6: Sonar-LCD

- Schreiben Sie ein Programm, das den aktuellen Abstandswert in Zoll in der oberen Zeile auf dem LC-Display anzeigt.
- Erweitern Sie das Programm, so dass in der unteren Zeile die Anzahl der empfangenen Sensorwerte hochgezählt wird.

Aufgabe 4.7: Sonar-PC

- Schreiben Sie ein Programm, das bei einer Änderung des Abstandswertes die Byte-Folge von Sonar-Sensor über USB an den PC sendet. Testen Sie das Programm mit HTerm.
- Für Forgeschrittenen, die sich bereits mit Oberflächenprogrammierung für den PC auskennen, Entwicklung eines Programms, dass den aktuellen Abstandswert in Zoll in einem Ausgabefenster auf dem PC anzeigt. Eine Anleitung zur Programmierung von Oberflächen mit Python ist u.a. zu finden auf:

  `http://techwww.in.tu-clausthal.de/site/Lehre/Schuelerinformatik[13]/`

  Hier finden Sie einen kompletten Python-Programmierkurs für Schüler.