Informatikwerkstatt, Foliensatz 13
Joystick und IR-Sensoren

G. Kemnitz

Institut für Informatik, TU Clausthal (IW-F13)

15. Dezember 2020

Inhalt:

- Joystick
- Infrarot-Abstandssensor
- Linienverfolgung
- Aufgaben

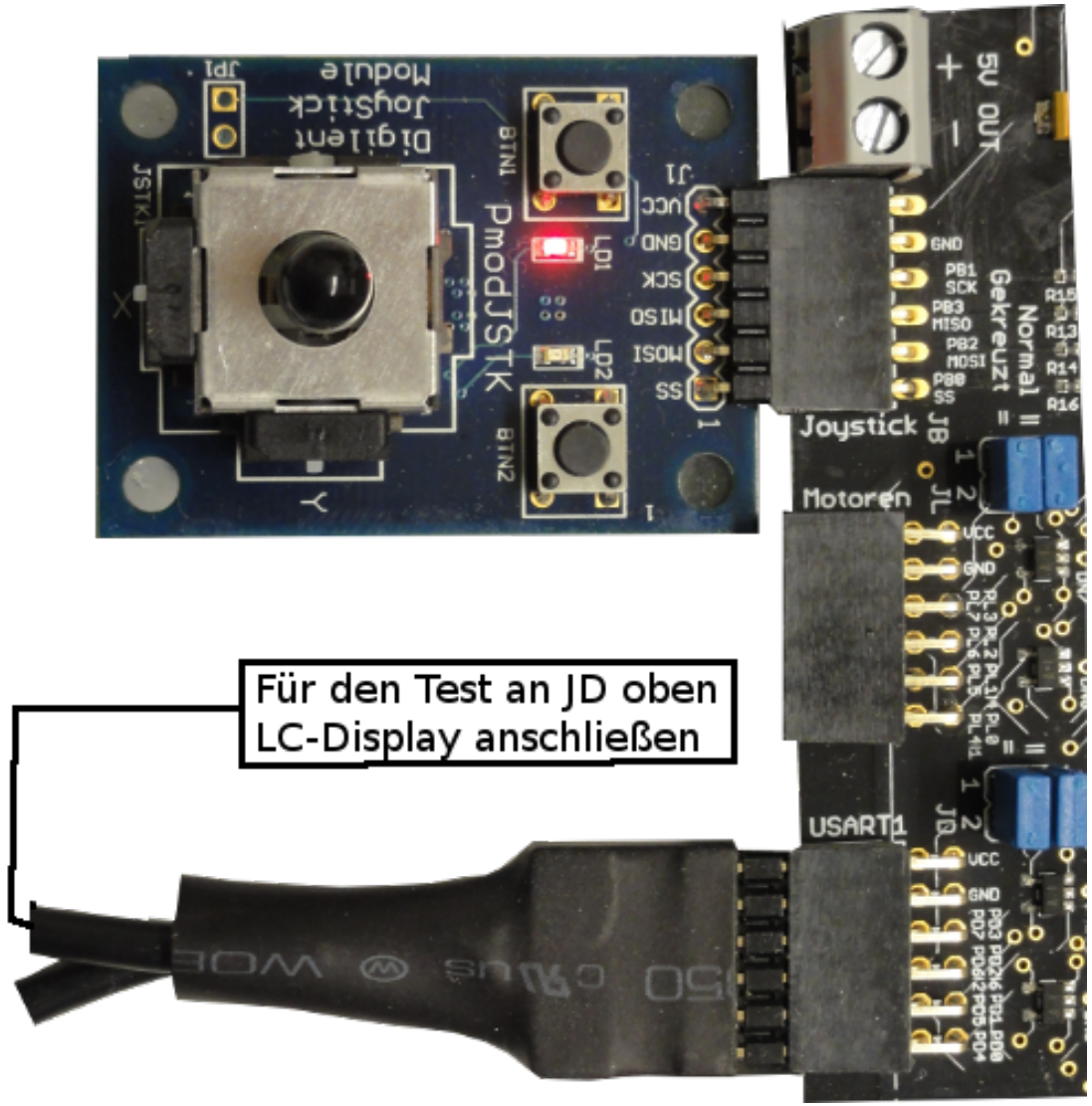
Interaktive Übungen:

1. Test des Joysticks (test_joystick).
2. Test des IR-Abstandssensors (test_sharpsens).
3. Test der Bodensensoren.

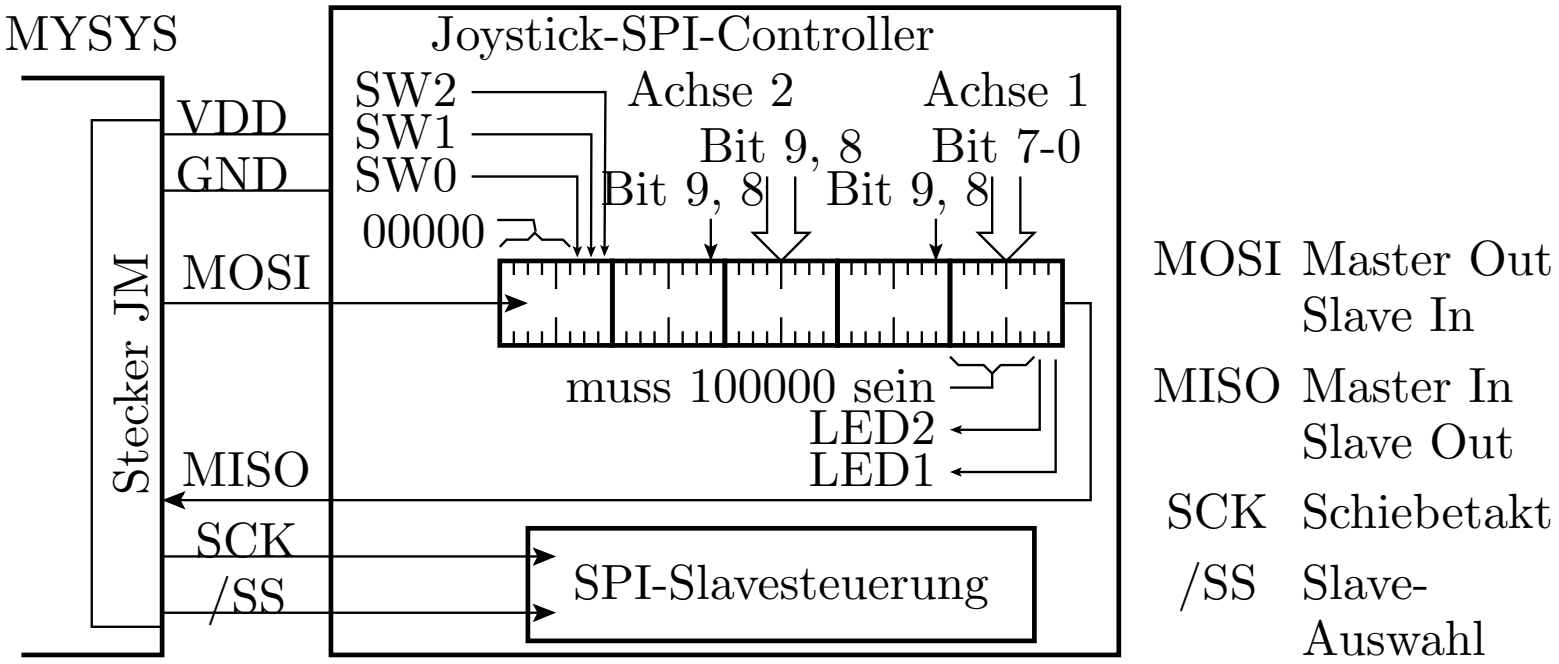
Joystick

Anschluss des Joysticks

- Joystick Pmod-JSTK an JB
- LCD PmodCLS an JD oben
- Kommunikation über SPI



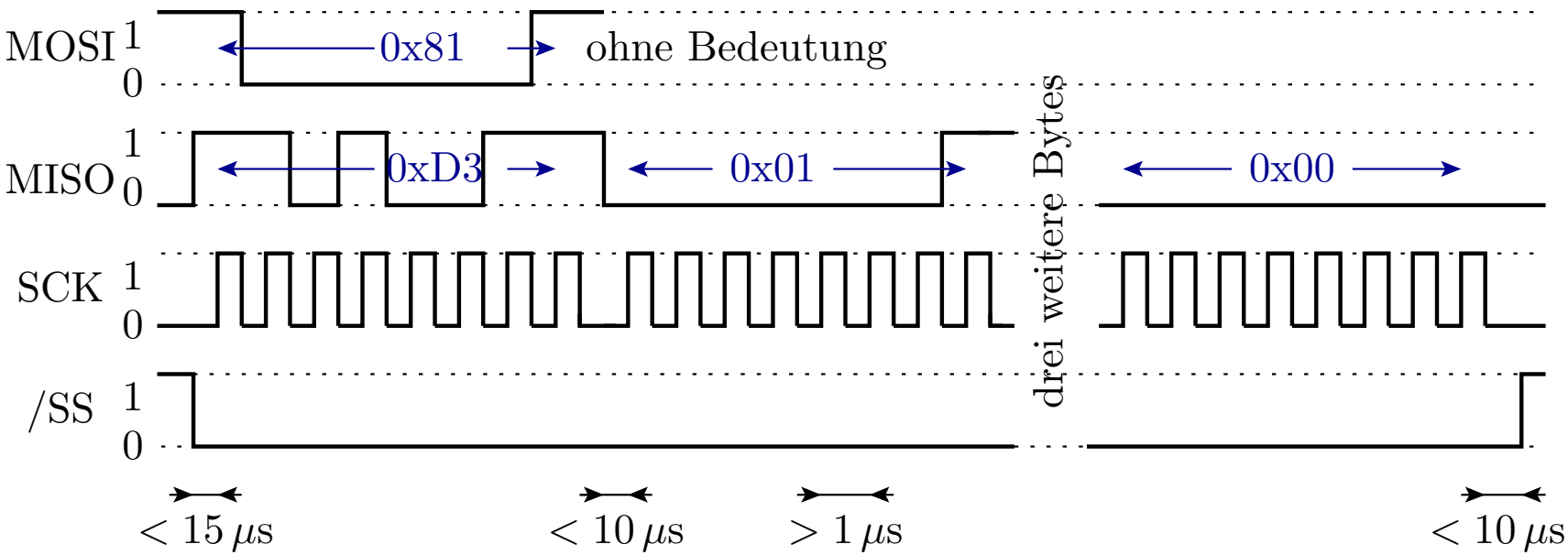
SPI-Funktionalität im Joystick[1]



- Der SPI-Slave des Joysticks funktioniert etwa wie ein 5x8 Bit Schieberegister, das vom Joystick Daten übernimmt, geschoben wird und Daten übergibt. Steuerung über SCK und /SS.
- Der Master im Mikrorechner muss /SS aktivieren, 5 Bytes schicken - warten - übernehmen und /SS deaktivieren.

[1] SPI – Serial Peripherial Bus

1. Joystick



- Die Zeiten im Bild sind einzuhalten, sonst Übertragungsfehler. Blockierendes serielles Versenden nach jedem Byte, Dauer bei 9600 Baud ca. $1\,\mathrm{ms} \gg 10\,\mu\mathrm{s}$, verursacht Fehlfunktion.
- Der Joystick übernimmt 0b100000$l_2 l_1$ (l_i – LED-Ausgabewert) + vier Bytes, die nicht ausgewertet werden, und
- sendet zwei Bytes mit dem x–Wert, zwei Bytes mit dem y–Wert und ein Byte mit drei Tasterwerten.

Initialisierung des SPI-Busses im Mikroprozessor

Die SPI-Initialisierung:

- Legt die Datenflussrichtung der SPI-Daten- und SPI-Steuersignale fest,
- deaktiviert das Slave-Auswahlsignal und
- Aktiveriert den SPI-Bus als Master mit Taktteiler 1/128:

```
void joystick_init(){
 DDRB = 0xf7;     // MOSI, SCK, /SS: Ausgänge
 PORTB |= 0x1;    // /SS=1 (Slave deaktiviert)
 //SPI als Master mit f_SCK=f_CPU/128 einschalten
 SPCR = (1<<SPE)|(1<<MSTR)|(0b11<<SPR0);
}
```

- Zur Nutzung im Interrupt-Modus wäre zusätzlich Interrupt-Bit »SPIE« für die lokale Freigabe zu setzen.

1. Joystick

Name	Address	Value	Bits	
SPCR	0x4C	0x53		
SPIE		0x00		Interrupt-Freigabe
SPE		0x01		SPI aktivieren
DORD		0x00		
MSTR		0x01		als Master
CPOL		0x00		
CPHA		0x00		
SPR		0x03		CPU-Takt/ 128
SPSR	0x4D	0x00		
SPIF		0x00		Ereignisbit
WCOL		0x00		
SPI2X		0x00		
SPDR	0x4E	0x00		

- Ereignisbit für die Abfrage »Übertragung fertig« ist »SPIF«

Datenaustausch mit dem Joystick

```
void joystick_get(uint8_t *buffer, uint8_t LedDat){
  PORTB &=~1;                          //Slave aktivieren
                                       //1. Byte senden
  SPDR = 0b10000000|(LedDat & 0b11);
  uint8_t i;
  for (i=0;i<5;i++){
    while(!(SPSR & (1<<SPIF)));//Warte SPI fertig
    buffer[i] = SPDR;
    if (i<4) SPDR = 0;                 //null senden
  }
  PORTB |= 1;                          //Slave deaktivieren
}
```

- 0x1000 00$l_1 l_0$ senden (l_i – Ausgabe an Led i).
- 4 Bytes enpfangen und 4$\times$null senden.
- Letztes Byte empfangen.

Testprogramm

- Konstanten für die LCD-Ausgabe:

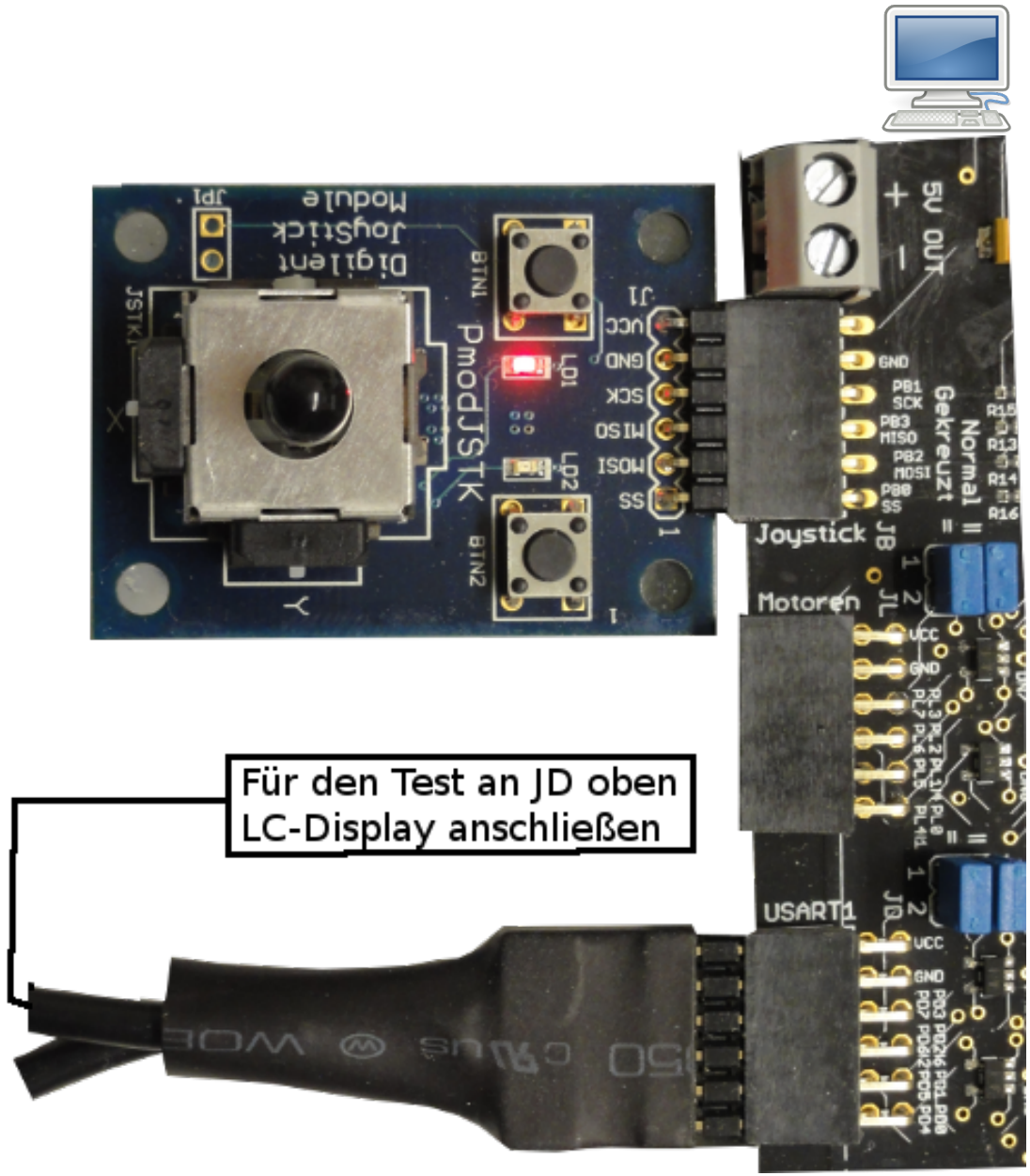
```
#define INITSTR "x=....␣Btn:␣xxx␣y=....␣Ct:......"
#define LCP_x      2 //x-Koordinate
#define LCP_y     18 //y-Koordinate
#define LCP_Btn1  14 //Button 1 gedrückt
#define LCP_Btn2  13 //Button 2 gedrückt
#define LCP_Btn3  12 //Button 3 gedrückt
#define LCP_Ct    26 //Zähler Übertragungen
```

1. Joystick

```
int main(void){
 joystick_init();                    //SPI init.
 lcd_init((uint8_t*)INITSTR);
 uint32_t Ct;                        //Datenzähler
 uint8_t dat[5];                     //Daten
 sei();                              //Interrupts ein
 while(1) {                          //Wiederhole immer
            //Lese Joystick-Daten + LED-Ausgabe
  joystick_get(dat, (Ct/200) & 0b11);
  lcd_disp_val((dat[1]<<8)+dat[0],LCP_x, 4);
  lcd_disp_val((dat[3]<<8)+dat[2],LCP_y, 4);
  if (dat[4] & 0b100) lcd_disp_chr('E', LCP_Btn3);
  else lcd_disp_chr('A', LCP_Btn3);
  if (dat[4] & 0b010) lcd_disp_chr('E', LCP_Btn2);
  else lcd_disp_chr('A', LCP_Btn2);
  if (dat[4] & 0b001) lcd_disp_chr('E', LCP_Btn1);
  else lcd_disp_chr('A', LCP_Btn1);
  lcd_disp_val(Ct++,LCP_Ct,6); //Zähler anzeigen
 }
}
```

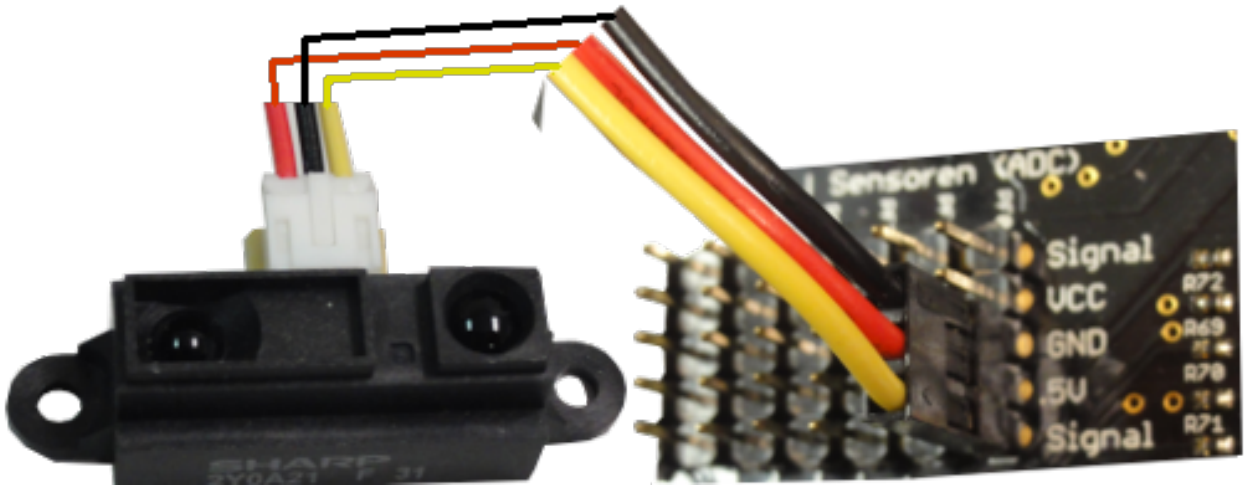
Joystick ausprobieren

- Joystick Pmod-JSTK an JB
- LCD PmodCLS an JD oben
- Projekt »F13-test_joystick\test_joystick« öffnen, übersetzen, starten.



Infrarot-Abstandssensor

Sensoranschluss an den Mikrorechner

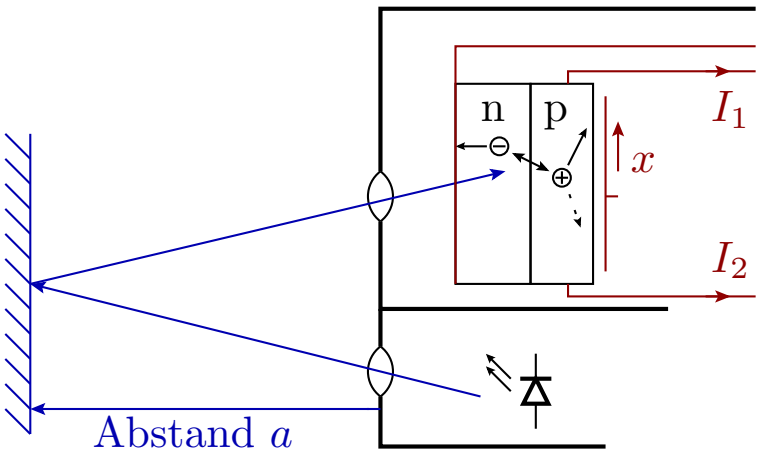


- Anschluss an den ADC-Stecker, ADC0 (PF0), 5 V-Seite.
- Zum Ausprobieren Multimeter Spannungsmessbereich zwischen Signal (gelb) und Masse ($\perp$) anschließen.
- In der IO-View nach Debug-Start eines beliebigen Programms PF0 und DDRF Bit 0 löschen.
- Hand im Abstand von 5 cm bis 50 cm vor dem Sensor bewegen. Bei ca. 7 cm Spannungsmaximum $> 3\,\mathrm{V}$.

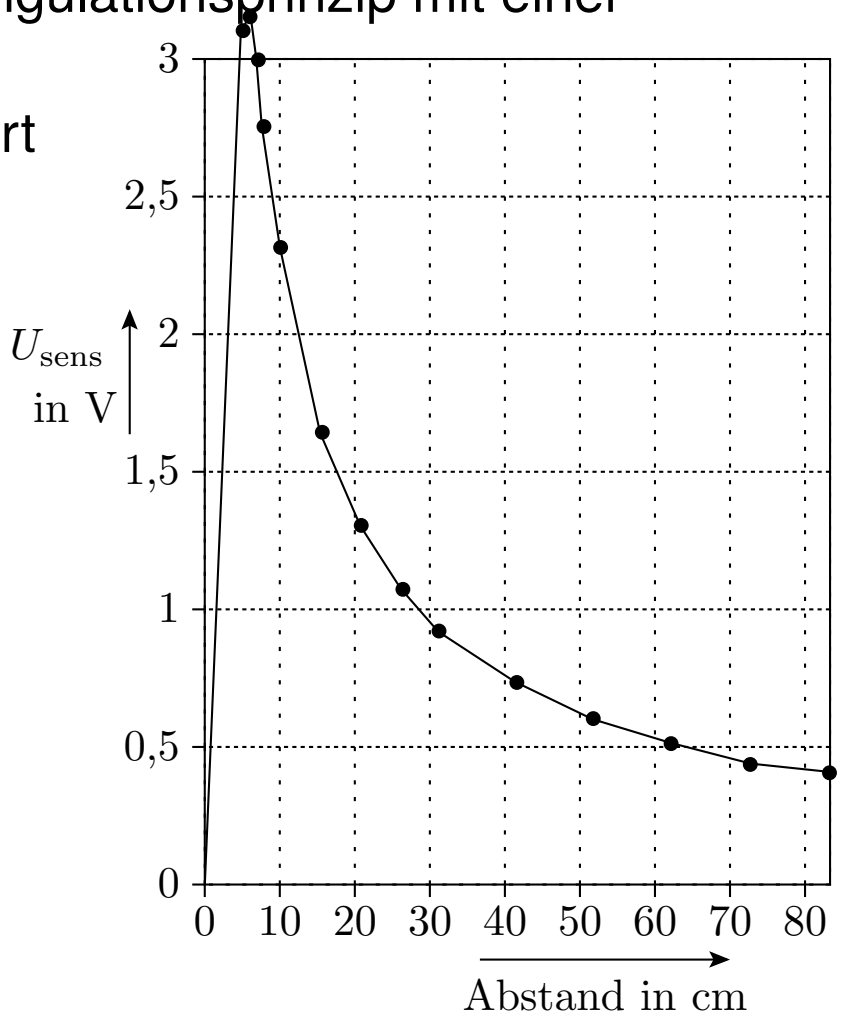
Infrarot-Abstandssensor Sharp 2Y0A21

Der Sensor arbeitet nach dem Triangulationsprinzip mit einer positionsempfindlichen Diode.

- Messspannung etwa umgekehrt proportional zum Abstand.
- Wandlungsgeschwindigkeit ca. 20 Werte pro Sekunde.



$$x = f(I_1, I_2) \text{ und } a \sim \frac{1}{x}$$



Initialisieren des Analog-Digital-Wandlers

Name	Address	Value	Bits
ADCSRA	0x7A	0x96	
ADEN		0x01	ADC einschalten
ADSC		0x00	
ADATE		0x00	
ADIF		0x01	Ereignisbit
ADIE		0x00	
ADPS		0x06	CPU-Takt / 64
ADCSRB	0x7B	0x00	
ADMUX	0x7C	0x40	
REFS		0x01	V_Ref = AVCC
ADLAR		0x00	
MUX		0x00	Kanal 0

- Der Sensor ist an ADC0 (PF0) (Kanal 0 auswählen).

- Der Wandlertakt als CPU-Takt durch Teilerwert

$$f_{\mathrm{ADC}} = \frac{f_{\mathrm{CPU}}}{64} \approx 117\,\mathrm{kHz}$$

soll im Bereich von 50 kHz bis 200 kHz liegen. Eine ADC-Wandlung dauert 13 Wandlertakte.

- Zur Vermeidung von Spannungsverfälschungen ist PF0 als Eingang mit Ausgabewert 0 (Pullup aus) einzustellen.

```
void adc_init(){
 ADMUX = 0b01000000;//Kanal 0 mit AREF = AVCC
     //Einschalten mit Wandlungstaktteiler 64
 ADCSRA = (1<<ADEN)|(0b110<<ADPS0);
 DDRF &= ~0x01;      //Sensoreingang als Eingang
 PORTF &=~0x01;      //Ausgabewert 0 (hochohmig)
}
```

- Wandlungsstart durch Setzen von ADSC in ADCSRA.
- Bei Wandlungsabschluss setzt der Prozessor ADIF=1.
- ADIF wird durch Schreiben einer Eins gelöscht.

Sensorwert als Vielfaches von 10µV

322 Wandlerergebnisse addierten[2]:

$$m = 322 \cdot 1024 \cdot \frac{U_{\text{sens}}}{3{,}3\,\text{V}} = U_{\text{sens}} \cdot 10^5 \text{V}^{-1}$$

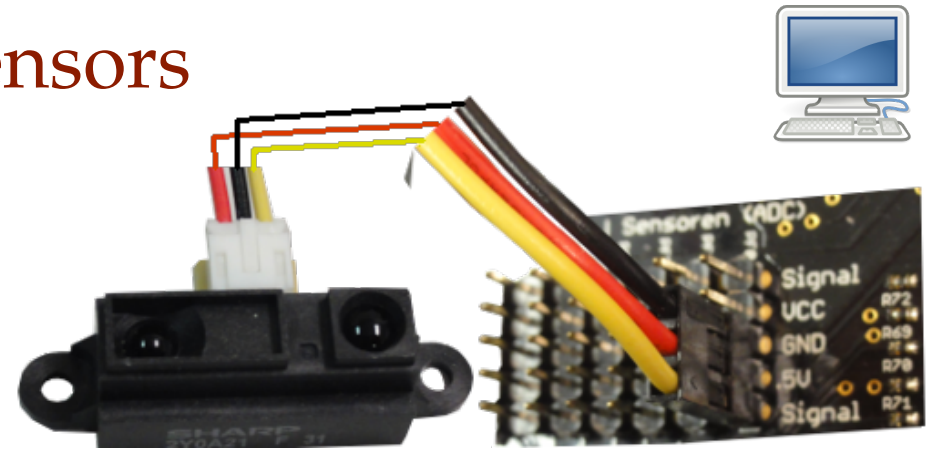
```
uint32_t get_adc(){
 uint16_t i;
 uint32_t wert=0;                    //Wert löschen
 for (i=0;i<322;i++){
  ADCSRA |= (1<<ADSC);               //Wandlung starten
  while(!(ADCSRA & (1<<ADIF)));//auf ADIF warten
  ADCSRA |= (1<<ADIF);               //ADIF löschen
  wert += ADC;                     //Ergebnisse addieren
 }
 return wert;
}
```

[2] Zunahmen der Standardabweisung: $\sqrt{322} \approx 18$. Verringert den relativen zufälligen Messfehler auf $\frac{\sqrt{322}}{322} \approx 6\%$ gegenüber »ohne Summation«.

Testrahmen für den Sensor mit LCD-Ausgabe


```
#include <avr/io.h>
#include "comir_lcd.h"
#include <avr/interrupt.h>
#define INITSTR "U_abst:␣....␣mV␣Ct:......␣␣␣␣␣␣␣"
#define LCP_UABST   8   //Sensorspannung
#define LCP_Ct     19   //Zähler
int main(void){
  adc_init();         //ADC Kanal 0  initialisieren
  lcd_init((uint8_t*)INITSTR);
  uint32_t dat, Ct=0;   //Messwert und Datenzähler
  sei();
  while(1) {
    dat = get_adc();
    lcd_disp_val(dat/100, LCP_UABST, 4);
    lcd_disp_val(Ct++,LCP_Ct, 6);
  }
}
```

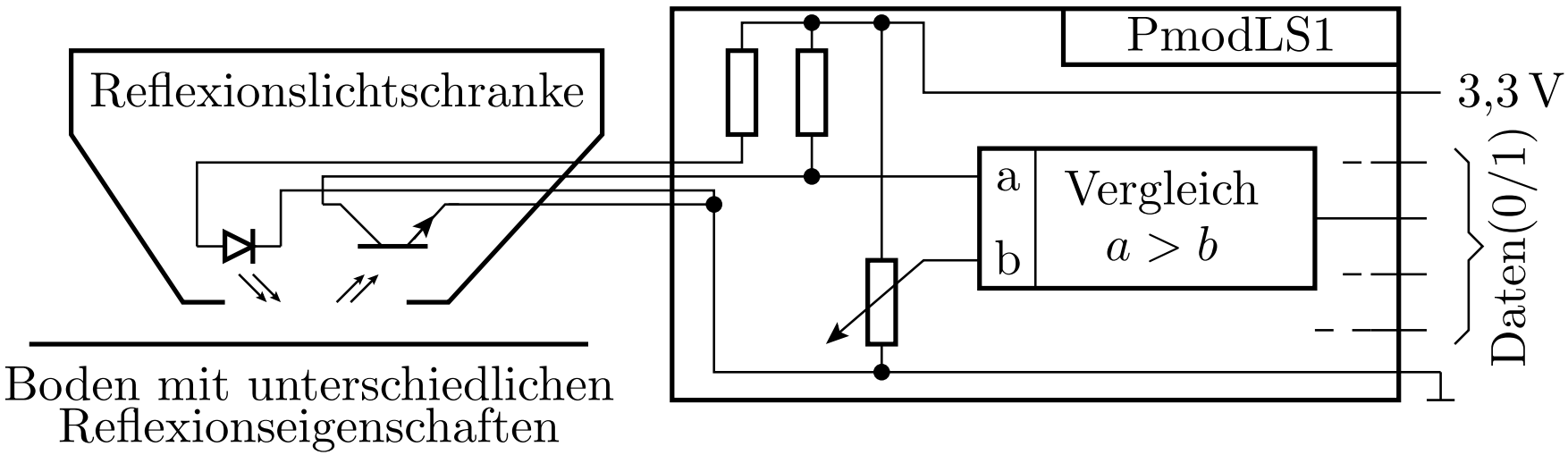
Test des IR-Abstandssensors



- LCD PmodCLS an JD oben.
- Sharp-Sensor an den ADC-Stecker, ADC0 (PF0), 5 V-Seite.
- Projekt »F13-test_sharpsens\sharpsens« öffnen, übersetzen, starten.
- Hand im Abstand von 5 cm bis 50 cm vor dem Sensor bewegen. Bei ca. 7 cm Spannungsmaximum $> 3\,\mathrm{V}$.

Linienverfolgung

Bodensensor



Der Bodensensor besteht aus vier Reflexionslichtschranken. Je näher oder besser reflektierend der Boden ist, desto stärker ist das Ausgangssignal des Fototransistors. Auf dem Liniensensormodul PmodLS1 befinden sich 4 Schwellwertschalter zur Wandlung in 1 für nahes Objekt und 0 sonst und 4 LEDs zur Anzeige des Wandlerergebnisses. Die Schaltschwelle wird mit dem Potentiometer auf dem PmodLS1 eingestellt.

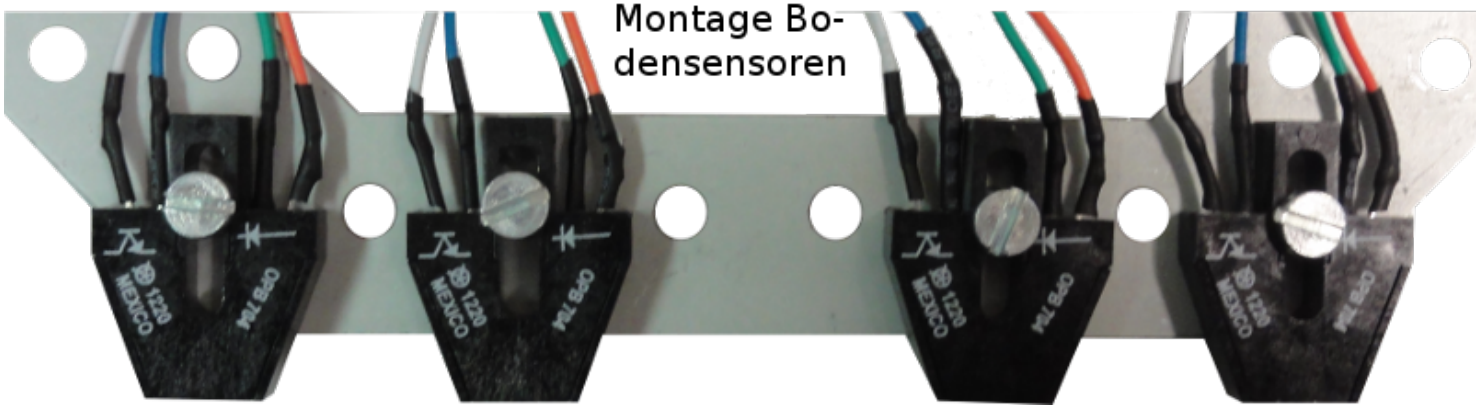
Anschluss der Sensoren

- vier Sensoren an PmodLS1 stecken.
- PmodLS1 an Stecker JK oben.
- Sensortest ohne Testprogramm mit den LEDs auf dem PmodLS1

Einstellung Ansprech-schwelle

Anschluss der Bodensen-soren über PmodLS1 an JK

Montage Bo-densensoren



Aufgaben

Aufgabe 13.1: Joystick-Treiber

Entwickeln Sie einen Joystick-Treiber:

- Der Treiber soll die Joystick-Daten mit einer ISR $\geq$20 mal je Sekunde auslesen und die LEDs aktualisieren.
- Die nachfolgenden Zugriffsfunktionen sollen nicht blockierend die zuletzt vom Joystick gelesenen Werte zurückgeben:

```
uint16_t js_get_x();      //x-Wert abholen
uint16_t js_get_y();      //y-Wert abholen
uint8_t  js_get_btn();    //Tasterwerte abholen
void js_set_led(uint8_t);//LED-Werte vorgeben
```

Erweitern Sie den Testrahmen auf Folie 11 so, dass alle Joystick-Treiberfunktionen eingebunden sind.

Aufgabe 13.2: IR-Abstandssensortreiber

Entwickeln Sie einen Treiber für den IR-Abstandssensor:

- Der ADC soll ständig Messen und die ADC-ISR soll immer 322 Messwerte summieren und die Summe abrufbereit halten.
- Funktion zur Ausgabe des Sensorsignal in mV:

```
uint16_t ir_get_abst_mV();
```

Testen Sie den Treiber mit dem Testrahmen auf Folie 19.

Aufgabe 13.3: Abstand in mm

- Untersuchen Sie, ob der Zusammenhang zwischen dem Abstand a und der Spannung U_{sens} durch eine Funktion

$$a = \frac{c_1}{U_{\text{sens}} - c_2}$$

  (c_1, c_2 – empirisch zu bestimmende Konstanten) oder stückweise linear angenähert werden kann.
- Entwickeln Sie einen Algorithmus, der die gemessene Spannung in einen Abstand in mm umrechnet.
- Erweitern Sie den Treiber für den IR-Abstandssensor um eine Funktion zur Rückgabe des Sensorabstands:

```
uint16_t ir_get_abst_mm();
```

- Erweiteren Sie den Testrahmen um eine Anzeige des Abstands.

Aufgabe 13.4: Linienverfolgung

- Überlegen Sie sich, wie die Bodensensoren zweckmäßig an das Fahrzeug angebaut werden müssen, um Linien auf dem Boden zu verfolgen.
- Testen Sie für unterschiedliche Linien (auf Papier, Fliesenfugen auf dem Flur, ...) ob der Sensor diese Linien erkennt und verfolgen kann.
- Entwickeln Sie eine Idee für einen Algorithmus für die Linienverfolgung und skizzieren Sie ihn als Programmablauf.
- Programmieren und testen Sie ihren Algorithmus unter Nutzung des Motortreibers und anderer Treiber.