



Informatikwerkstatt, Foliensatz 7

Joystick und IR-Sensoren

G. Kemnitz

Institut für Informatik, TU Clausthal (IW-F7)
11. Januar 2016



Inhalt Foliensatz 7

Joystick

Infrarot-Abstandssensor

Linienverfolgung

Aufgaben

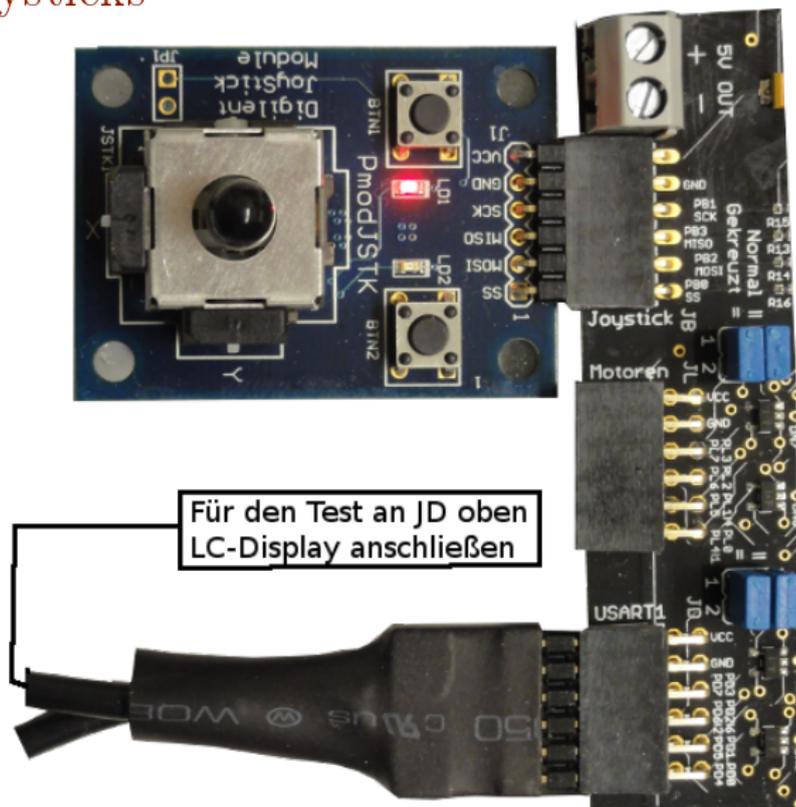


Joystick



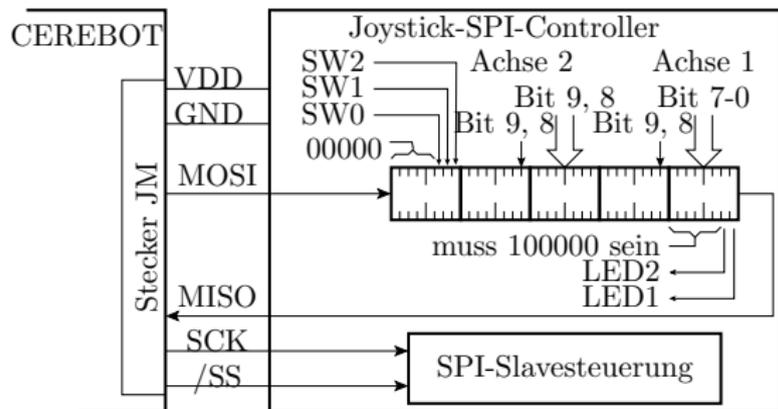
Anschluss des Joysticks

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





SPI-Funktionalität im Joystick¹



Slave-Steuerung

Aktion	/SS	SCK
Übernahme		
Schieben	0	
Ausgabe		
keine		sonst

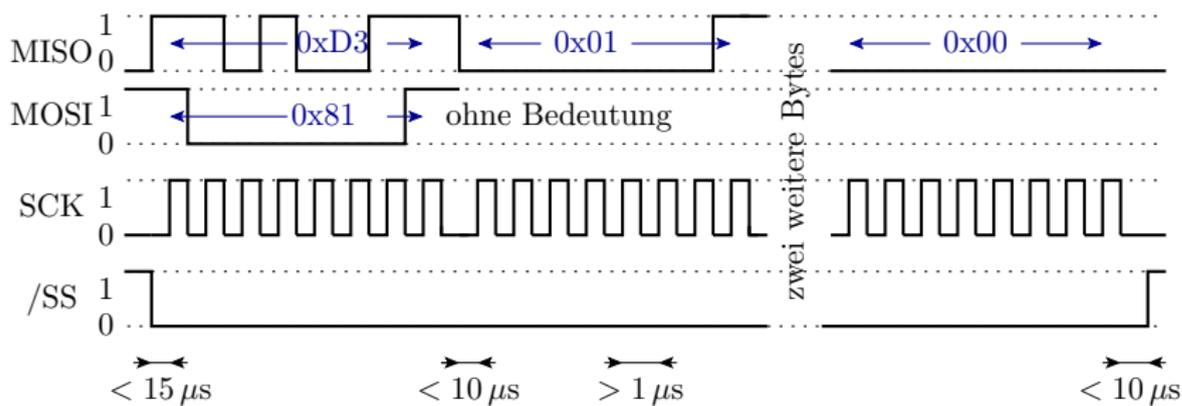
MOSI Master Out Slave In
MISO Master In Slave Out
SCK Schiebetakt
/SS Slave-Auswahl

- Den SPI-Slave des Joysticks kann man sich wie ein 5x8 Bit Schieberegister vorstellen, 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.

¹SPI – Serial Peripheral Bus



1. Joystick



- Die Zeiten im Bild sind einzuhalten, sonst Übertragungsfehler. Eine sofortige serielle Weitergabe nach Byteempfang verboten, da Sendedauer bei 9600 Baud ca. $1 \text{ ms} \gg 10 \mu\text{s}$ max. zulässige Pause zwischen zwei SPI-Übertragungen.
- Gesendet wird ein Byte $0b100000l_2l_1$ (l_i – LED-Ausgabewert) und 4×beliebige Bytes.
- Empfangen werden zwei Bytes mit dem x -Wert zwei Bytes mit dem y -Wert und eines 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
- Aktiviert den SPI-Bus als Master mit Taktteiler 1/128:

```
void joystick_init(){  
    DDRB = 0xf7;    // /SS, MOSI, SCK: 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 Interruptmodus wäre zusätzlich Interruptbit »SPIE« für die lokale Freigabe zu setzen.



1. Joystick

Name	Address	Value	Bits
[-] SPCR	0x4C	0x53	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
SPIE	0x00		<input type="checkbox"/> Interrupt-Freigabe
SPE	0x01		<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> SPI aktivieren
DORD	0x00		<input type="checkbox"/>
MSTR	0x01		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> als Master
CPOL	0x00		<input type="checkbox"/>
CPHA	0x00		<input type="checkbox"/>
SPR	0x03		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> CPU-Takt/ 128
[-] SPSR	0x4D	0x00	<input type="checkbox"/>
SPIF	0x00		<input type="checkbox"/> Ereignisbit
WCOL	0x00		<input type="checkbox"/>
SPI2X	0x00		<input type="checkbox"/>
SPDR	0x4E	0x00	<input type="checkbox"/>

- 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
}
```

- $0x100000l_1l_0$ senden (l_i – Ausgabe an Led i)
- 4 Bytes empfangen 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
    }
}
```





Entwickeln des Treiber (selbst beenden)



- Der Treiber soll die Joystick-Daten mit einer ISR mindestens 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_btn1(); // Wert Btn1 abholen
uint8_t js_get_btn2(); // Wert Btn2 abholen
uint8_t js_get_btn3(); // Wert Btn3 abholen
```

- Der mit

```
void js_set_led(uint8_t Led);
```

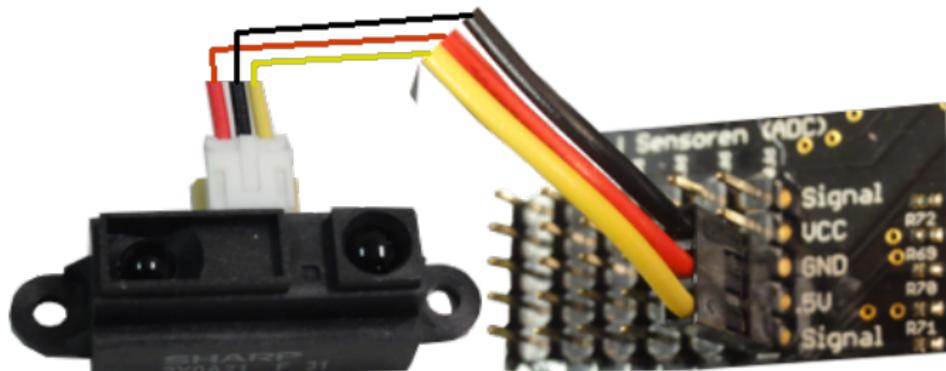
übergebene Wert soll nicht blockierend zwischengespeichert und spätestens 50 ms später angezeigt werden.

- Die 2-Byte-Leseoperationen der x- und y-Werte in den js_get-Funktionen dürfen nicht unterbrechbar sein.



Infrarot-Abstandssensor

Sensoranschluss an den Mikrorechner

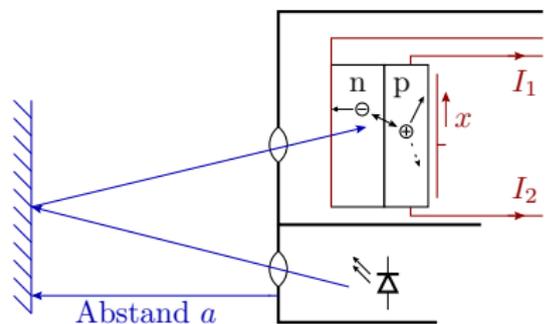


- Anschluss an den ADC-Stecker, ADC0 (PF0), 5 V-Seite.
- Zum Ausprobieren Multimeter Spannungsmessbereich zwischen Signal (gelb) und Masse (\perp) anschließen.
- In der IO-View nach Debug-Start eines beliebigen Programms PORTF.0 und DDRF.0 löschen.
- Hand im Abstand von 5 cm bis 50 cm vor dem Sensor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

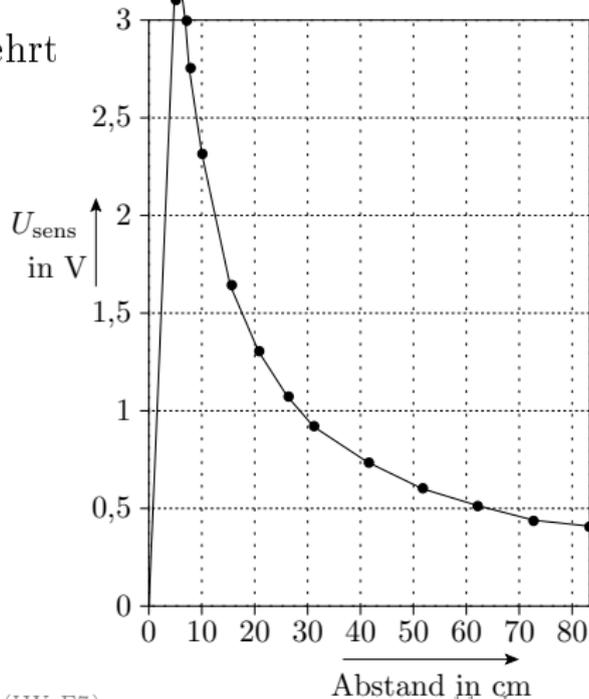
Infrarot-Abstandssensor Sharp 2Y0A21

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	
<small>ADC DAC</small> ADCSRA	0x7A	0x96	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	
<small>ADC DAC</small> ADEN	0x01	0x01	<input checked="" type="checkbox"/> <input type="checkbox"/>	ADC einschalten
<small>ADC DAC</small> ADSC	0x0C	0x0C	<input type="checkbox"/>	
<small>ADC DAC</small> ADATE	0x0C	0x0C	<input type="checkbox"/>	
<small>ADC DAC</small> ADIF	0x01	0x01	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Ereignisbit
<small>ADC DAC</small> ADIE	0x0C	0x0C	<input type="checkbox"/>	
<small>ADC DAC</small> ADPS	0x06	0x06	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	CPU-Takt / 64
<small>ADC DAC</small> ADCSRB	0x7B	0x00	<input type="checkbox"/>	
<small>ADC DAC</small> ADMUX	0x7C	0x40	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
REFS	0x01	0x01	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	V_Ref = AVCC
ADLAR	0x0C	0x0C	<input type="checkbox"/>	
<small>ADC DAC</small> MUX	0x0C	0x0C	<input type="checkbox"/>	Kanal 0

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



2. Infrarot-Abstandssensor

- Der Wandlertakt als CPU-Takt durch Teilerwert

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

soll im Bereich von 500 bis 200 kHz liegen. Eine ADC-Wandlung dauert 13 Wandlertakte.

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

```
void adc_init(){
    ADMUX = 0b01000000; // Kanal 0 mit AREF = AVCC
    // Einschalten mit Wandlungstakteiler 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 $10\mu\text{s}$

- 322 Wandlerergebnisse addierten²:

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

```
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;                // Wandlerwerte add.
    }
    return wert;
}
```

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



Entwickeln des Treiber (selbst zu Ende)



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

```
uint16_t sharpsens_mV();
```

- Funktion zur Rückgabe des Sensorabstands in Millimeter³:

```
uint16_t sharpsens_mm();
```

³Für die Umrechnung könnte man zwischen den Messpunkten linear interpolieren oder es mit einer $1/x$ -Funktion versuchen. Sonderbehandlung für »außerhalb des Messbereichen«, z.B. dann Rückgabe des größten darstellbaren Wertes oder null.



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

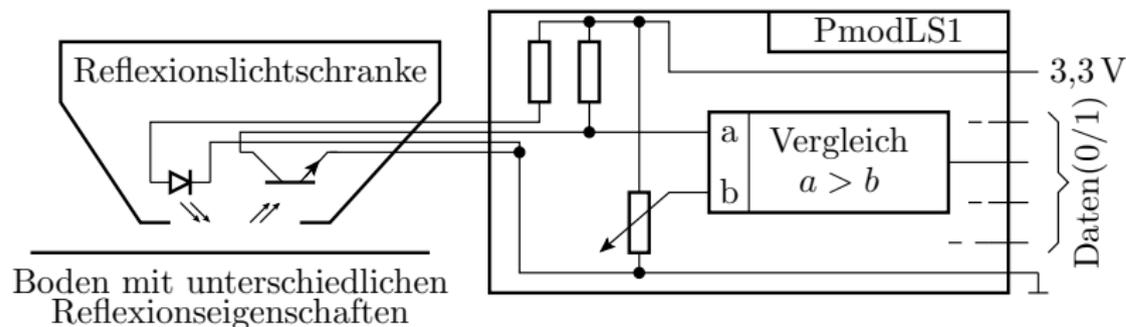
int main(void){
    adc_init(); // ADC Kanal 0 initialisieren
    lcd_init((uint8_t*)INITSTR);
    uint32_t dat, Ct=0; // Messwert und Datenzähler
    sei();
    while(1) {
        dat = get_adc();
        lcd_disp_val(dat/100, LCP_UABST, 4);
        lcd_disp_val(Ct++,LCP_Ct, 6);
    }
}
```





Linienverfolgung

Bodensensor



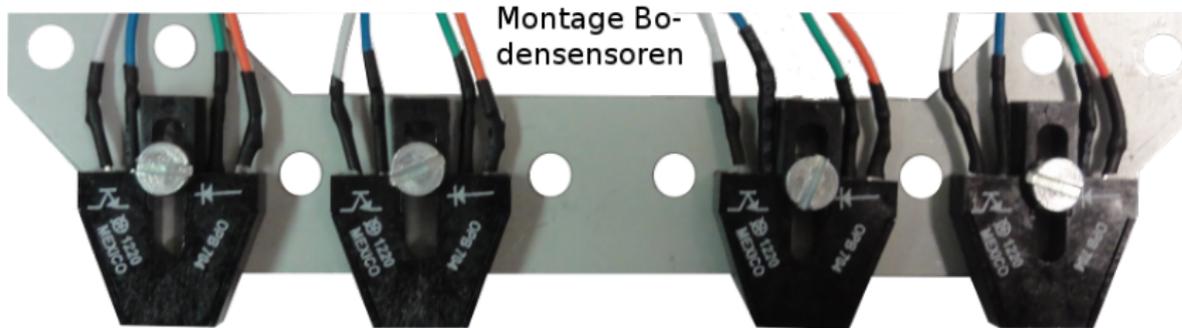
Der Bodensensor besteht aus vier Reflexionslichtschranken. Je näher oder besser reflektierend der Boden ist, desto stärker ist das Ausgangssignal des Fototransistors. Auf dem Liniensensormodul PmodLS1 befinden sich 4 Schwellwertschalter zur Wandlung in 1 für nahes Objekt und 0 sonst und 4 LEDs zur Anzeige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





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.



Aufgaben



Aufgabe 7.1: Joystick-Treiber



- Schreiben Sie die Programmdatei und den Header für den Joystick-Treiber.
- Ändern Sie den vorgegebenen Testrahmen so ab, dass alle Joystick-Treiberfunktionen eingebunden sind.



Aufgabe 7.2: IR-Abstandssensortreiber



- Schreiben Sie die Programmdatei und den Header für den IR-Abstandssensortreiber.
- Ändern Sie den vorgegebenen Testrahmen so ab, dass alle Treiberfunktionen eingebunden sind.
- Testen 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) angenähert werden kann.



Aufgabe 7.3: Linienverfolgung



Programmieren Sie Ihren Linienverfolgungsalgorithmus unter Nutzung des Motor-Treibers und testen Sie ihn.