

1 Kannlisten Aufgaben

Im Folgenden werden die Lösungen zu den Aufgaben aus der Kannliste dargestellt (siehe Anhang, S. 34).

Da ich seit Donnerstag krank bin, war es mir leider nicht möglich, einige Aufgaben bis zur Abgabefrist am Sonntag um 20 Uhr zu bearbeiten.

1.1 Aufgaben 0 bis 5

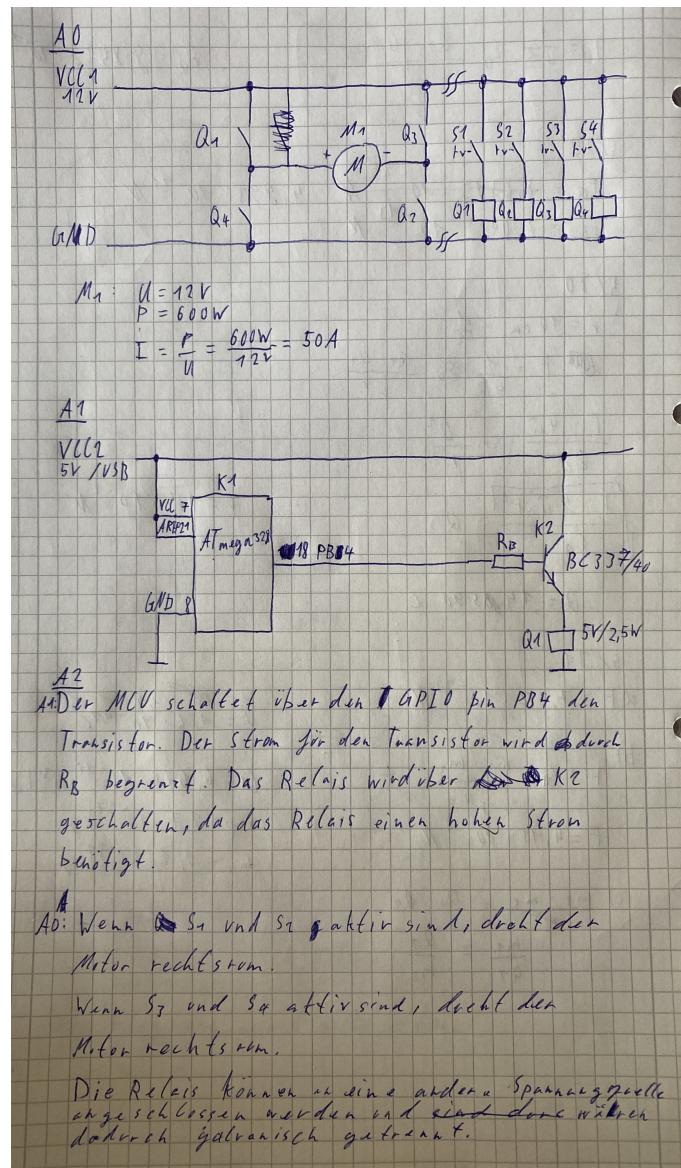


Abbildung 1: Aufgaben 0 bis 2

A3

geg.

$$U = 5V$$

$$P_{Q1} = 2,5W$$

$$U_{BE} = 0,85V$$

$$R_B = 350\Omega$$

$$U_{CEsat} = 0,7V$$

~~Übersetzung~~

$$U_{Hmin} = 2,4V$$

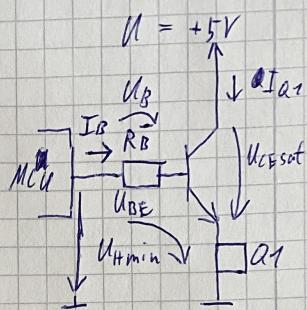
$$I_{Hmax} = 10mA$$

ges.

$$R_B$$

Aufforderungen:

$$I_{BB} < I_{Hmax}$$



~~$$P_{Q1} = \frac{U_{Q1}^2}{R_{Q1}} = \frac{(5V)^2}{2,5k\Omega} = 10W$$~~

~~$$P_{Q1} = \frac{U_{Q1}^2}{R_{Q1}} = \frac{5V \cdot 0,7V}{10\Omega} = 0,35W$$~~

$$R_{Q1} = \frac{U_{Q1}^2}{P_{Q1}} = \frac{(5V)^2}{2,5W} = 10\Omega$$

für $U_{Q1} = U - U_{CEsat}$:

$$I_{Q1} = \frac{U_{Q1}}{R_{Q1}} = \frac{5V - 0,7V}{10\Omega} = 430mA$$

$$I_B = \frac{I_{Q1}}{\beta \cdot \alpha} = \frac{430mA}{8 \cdot 0,5} = \frac{430mA}{175} = 2,46mA < 10mA$$

$$\begin{aligned} P_{Q1} &= U_{Q1} \cdot I \\ \Rightarrow P_{Q1} &= \frac{U_{Q1}^2}{R_{Q1}} \\ \Rightarrow R_{Q1} &= \frac{U_{Q1}^2}{P_{Q1}} \end{aligned}$$

$$\alpha = 0,5$$

$\alpha \hat{=} \text{doppelte Übersteuerung}$

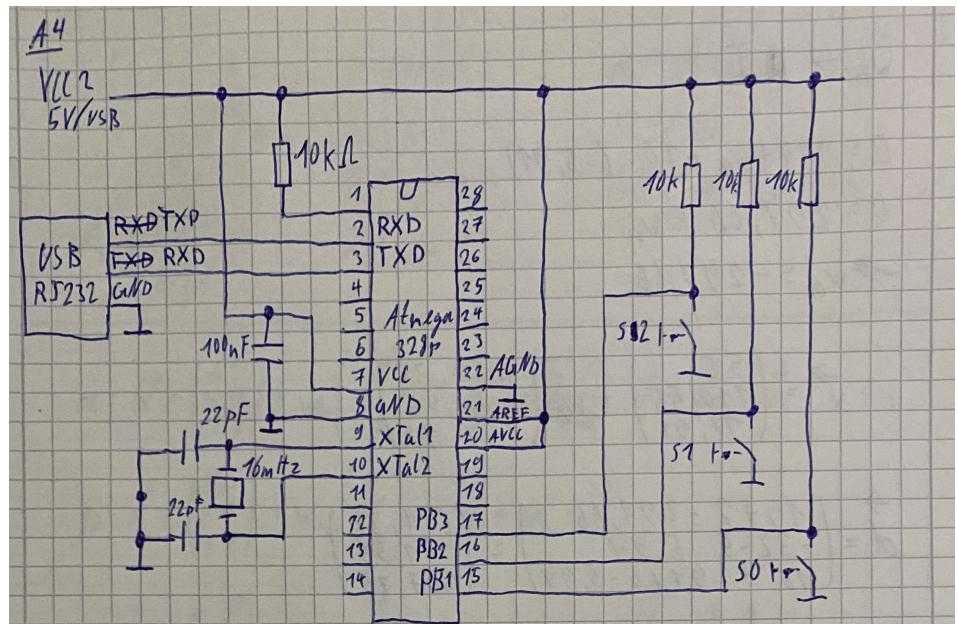
$$R_B = \frac{U_{Hmin} - U_{BE}}{I_B} \approx 630\Omega = \frac{2,4V - 0,85V}{2,46mA}$$

$$R_{Bmin} = \frac{U}{I_{Hmax}} = \frac{5V}{10mA} = 500\Omega$$

E12 Reihe: $\underline{680\Omega} \approx 630\Omega \geq 500\Omega$

Ich entscheide mich für $R_B = 680\Omega$, durch die Übersteuerung ist der Strom nicht zu niedrig.

Abbildung 2: Aufgabe 3



- 16MHz Quarz zwischen XTAL1 (9) und XTAL2 (10).
- 22pF Kondensatoren zwischen jeweils Pin 9 und 10 und Ground.
- 10kΩ Widerstand zwischen 5V und Reset (1)
- 100nF Kondensator zwischen 5V und Ground des IC.
- AREF auf 5V
- RS232 an Serielle Kommunikation anschließen.

A5

(gezeichnet in A4)

Im Grundzustand liegt der jeweilige Pin auf High. Der 10kΩ Widerstand schützt vor einem Kurzschluss. Wenn ein Taster betätigt wird, liegt der MCV Pin auf Low.

Abbildung 3: Aufgabe 4 und 5

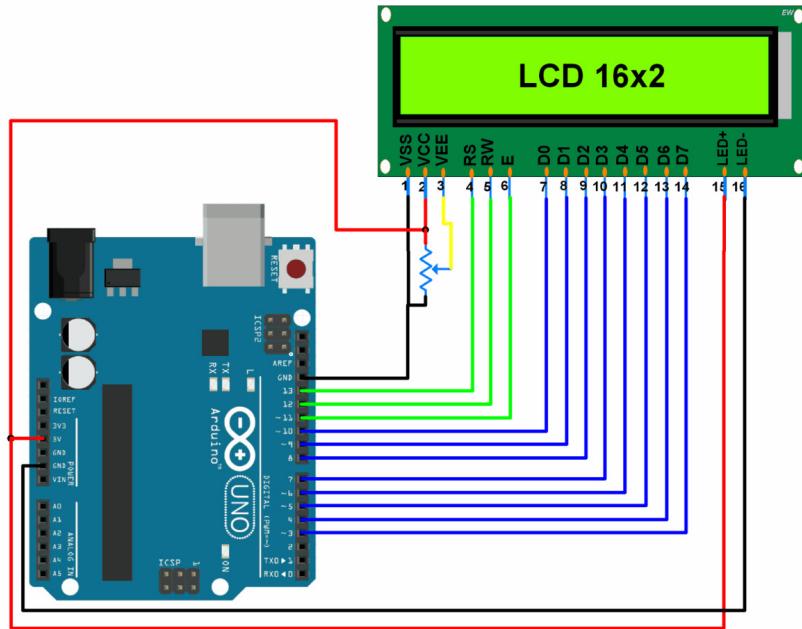


Abbildung 4: Schaltplan um LCD (mit HD44780) an Arduino anzuschließen

1.2 Aufgabe 6

In Abbildung 4 wie man ein LCD an einen Arduino anschließen kann. Die folgenden Pinbeschreibungen wurden aus dem Datenblatt des HD44780 entnommen.

Pin	MCU Pin	Funktionen der Ausgänge des HD44780
VSS	GND	niedriges Spannungspotential
VCC	5 V	positive Spannung für das LCD und den HD44780
VEE	[0 V; 5 V]	Kontrast – stellt ein wie stark sich die Kristalle drehen.
RS	PB5	0: Instruction 1: Data register
R/W	5 V	In diesem Fall wird nur geschrieben.
E	PB3	Enable – an ihm wird das Taktsignal angelegt.
D0 - D2	PB2 - PB0	8 Bit Dateneingang
D3 - D7	PD7 - PD3	8 Bit Dateneingang
LED+	5 V	Zuständig für die Spannungsversorgung der LEDs. Dabei ist darauf zu achten, dass ein Vorwiderstand in dem LCD Modul vorhanden ist.
LED-	GND	Hier muss das niedrige Spannungspotential angelegt werden.

1.3 Aufgabe 7

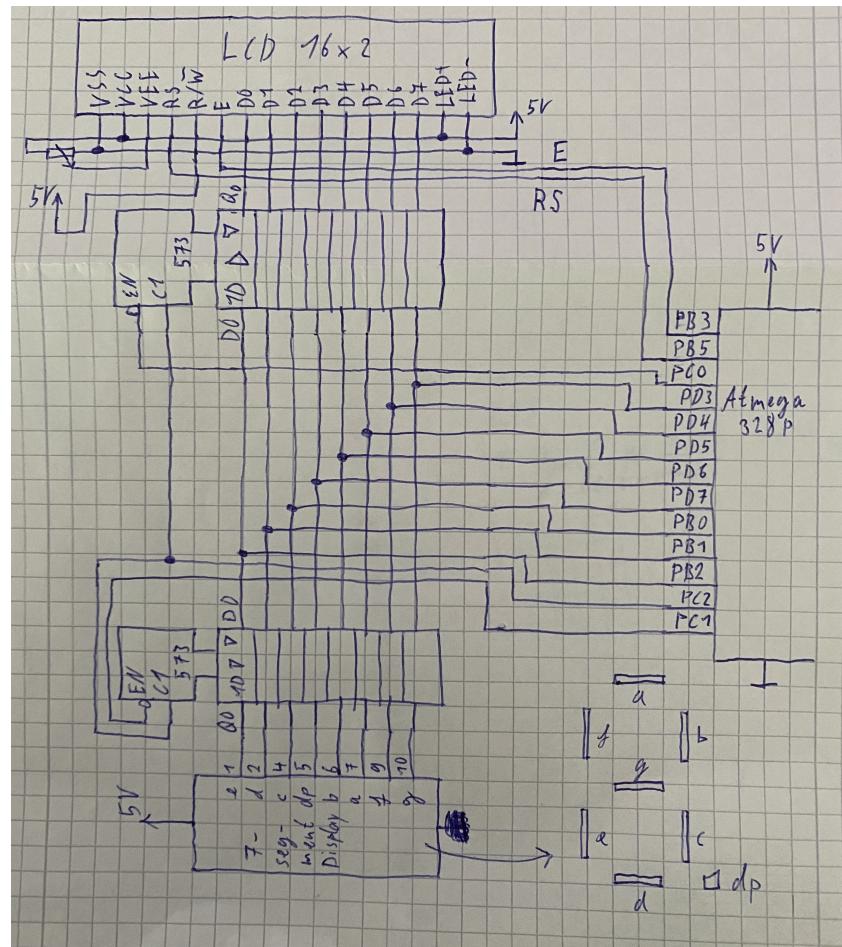


Abbildung 5: Schaltplan um LCD (mit HD44780) an Arduino anzuschließen

1.4 Allgemeiner Code

Code der für alle Aufgaben von 8 bis 14 gilt:

```
1  /*
2   * Warnung: Boolische True-Werte müssen den Wert 1 haben
3   *
4   * Zuweisungsliste:
5   *
6   * Arduino | MCU Pin | Bauteil Pin | Bezeichnung
7   * 5V      Vcc
8   * GND     Vss
9   * 13      PB5      RS          LCD
10  * 12      PB4      R/~W       LCD
11  * 11      PB3      E           LCD
12  * 10      PB2      D0          8 bit databus
13  * 9       PB1      D1          8 bit databus
14  * 8       PB0      D2          8 bit databus
15  * 7       PD7      D3          8 bit databus
16  * 6       PD6      D4          8 bit databus
17  * 5       PD5      D5          8 bit databus
18  * 4       PD4      D6          8 bit databus
19  * 3       PD3      D7          8 bit databus
20  * 2       PD2      CLK         Datenübernehmen für 7 segment display
21  * A0      PC0      S0          Taster in Negativlogik
22  * A1      PC1      S1          Taster in Negativlogik
23  * A2      PC2      S2          Taster in Negativlogik
24  * A3      PC3      Q1          Relais Rechts
25  * A4      PC4      Q2          Relais Links
26  * A5      PC5
27  * */
28
29 #define PIN_S0_AUS PC0
30 #define PIN_S1_RECHTS PC1
31 #define PIN_S2_LINKS PC2
32 #define PIN_Q1_RECHTS PC3
33 #define PIN_Q2_LINKS PC4
34
35 #define F_CPU 16000000
```

1.5 Aufgabe 8

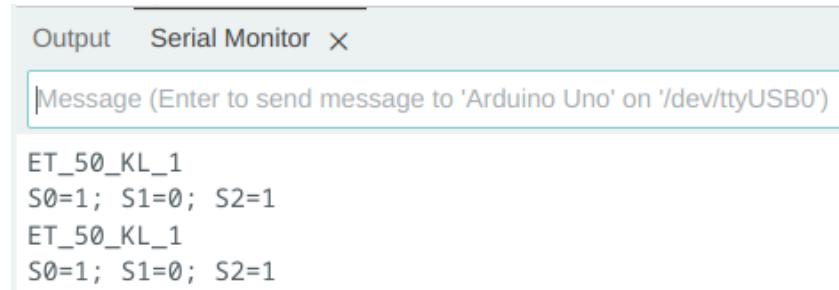


Abbildung 6: UART-Ausgabe von ET_50_KL_01

```
1 #define INFO "ET_50_KL_1\r\n"
2 #define BAUD 19200UL
3
4 #include "uartAT328p.h"
5 #include <stdbool.h>
6
7 int main(void) {
8     // GPIO Ausgänge setzen
9     DDRB = 0x3F;
10    DDRC = 0x18;
11    DDRD = 0xFC;
12
13    // UART initialisieren
14    usart_init();
15
16    // Version über UART senden
17    usart_puts(INFO);
18
19    // Port Eingänge in Variablen speichern
20    // Taster S0 bis S2
21    bool s0_aus = !((PINC >> PIN_S0_AUS) & 0x01);
22    bool s1_rechts = !((PINC >> PIN_S1_RECHTS) & 0x01);
23    bool s2_links = !((PINC >> PIN_S2_LINKS) & 0x01);
24
25    // Taster Zustände senden
26    usart_puts("S0=");
27    usart_itoa(s0_aus);
28    usart_puts("; S1=");
29    usart_itoa(s1_rechts);
30    usart_puts("; S2=");
31    usart_itoa(s2_links);
32    usart_puts("\r\n");
33 }
```

1.6 Aufgabe 9

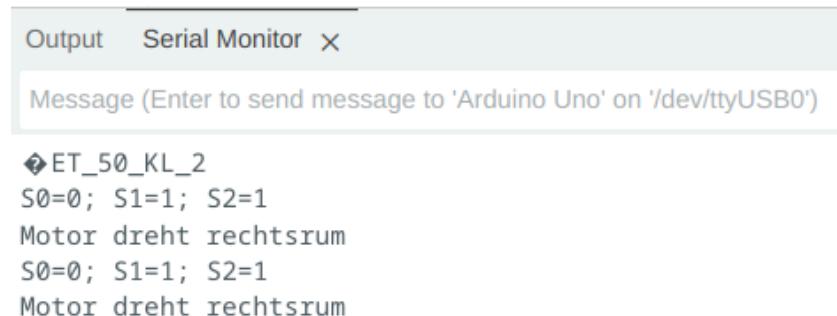


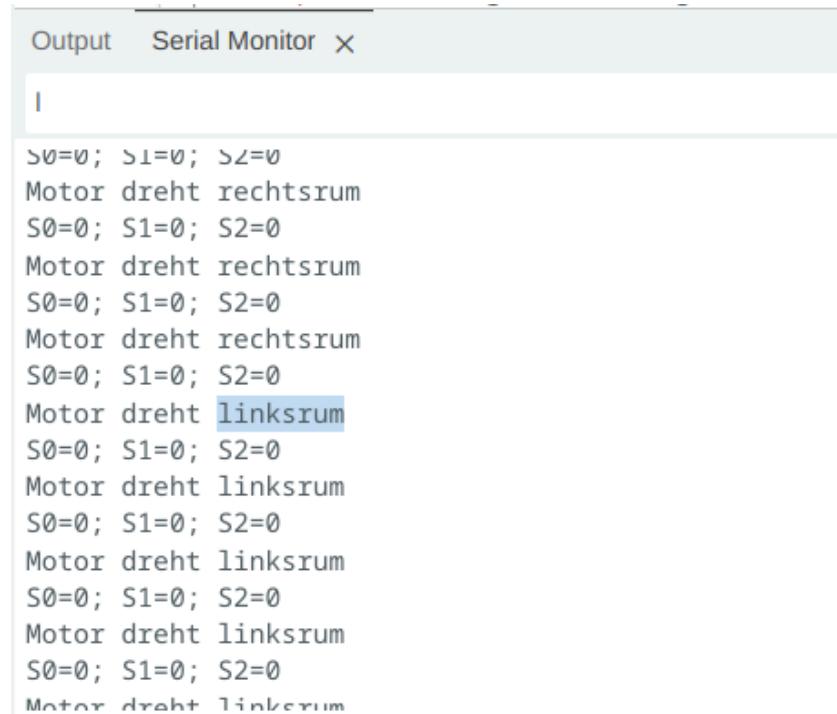
Abbildung 7: UART-Ausgabe von ET_50_KL_02

```
1 #define INFO "ET_50_KL_2\r\n"
2 #define BAUD 19200UL
3
4 #include "uartAT328p.h"
5 #include <stdbool.h>
6
7 // Relais Zustand
8 bool q1_rechts = 0;
9 bool q2_links = 0;
10
11 int main(void) {
12     // GPIO Ausgänge setzen
13     DDRB = 0x3F;
14     DDRC = 0x18;
15     DDRD = 0xFC;
16
17     // UART initialisieren
18     usart_init();
19
20     // Version über UART senden
21     usart_puts(INFO);
22
23     while (1) {
24         // Port Eingänge in Variablen speichern
25         // Taster S0 bis S2 (1=gedrückt)
26         bool s0_aus = !((PINC >> PIN_S0_AUS) & 0x01);
27         bool s1_rechts = !((PINC >> PIN_S1_RECHTS) & 0x01);
28         bool s2_links = !((PINC >> PIN_S2_LINKS) & 0x01);
29
30         if (s0_aus) {
31             // Aus
32             q1_rechts = 0;
33             q2_links = 0;
34         } else if (s1_rechts) {
```

```

35     // Rechtsrum
36     q1_rechts = 1;
37     q2_links = 0;
38 } else if (s2_links) {
39     // Linksrsum
40     q1_rechts = 0;
41     q2_links = 1;
42 }
43
44 // Taster Zustände senden
45 usart_puts("S0=");
46 usart_itoa(s0_aus);
47 usart_puts("; S1=");
48 usart_itoa(s1_rechts);
49 usart_puts("; S2=");
50 usart_itoa(s2_links);
51 usart_puts("\r\n");
52
53 if (q1_rechts) {
54     usart_puts("Motor dreht rechtsrum\r\n");
55 } else if (q2_links) {
56     usart_puts("Motor dreht linksrsum\r\n");
57 }
58
59 // Alle Relais Pins (PC3 und PC4) werden geleert und dann gesetzt
60 PORTC = PORTC & ~(1 << PIN_Q1_RECHTS | 1 << PIN_Q2_LINKS) |
61     q1_rechts << PIN_Q1_RECHTS | q2_links << PIN_Q2_LINKS;
62 }
63 }
```

1.7 Aufgabe 10



The screenshot shows the Arduino Serial Monitor window. The title bar has tabs for "Output" and "Serial Monitor". The main area of the window displays the following text, which is a repeating sequence of motor control messages:

```
S0=0; S1=0; S2=0
Motor dreht rechtsrum
S0=0; S1=0; S2=0
Motor dreht rechtsrum
S0=0; S1=0; S2=0
Motor dreht rechtsrum
S0=0; S1=0; S2=0
Motor dreht linksrum
```

Abbildung 8: UART-Ausgabe von ET_50_KL_03

```
1 #define INFO "ET_50_KL_3\r\n"
2 #define BAUD 19200UL
3
4 #include "uartAT328p.h"
5 #include <stdbool.h>
6
7 // Relais Zustand
8 bool q1_rechts = 0;
9 bool q2_links = 0;
10
11 int main(void) {
12     // GPIO Ausgänge setzen
13     DDRB = 0x3F;
14     DDRC = 0x18;
15     DDRD = 0xFC;
16
17     // UART initialisieren
18     usart_init();
19
20     // Version über UART senden
21     usart_puts(INFO);
```

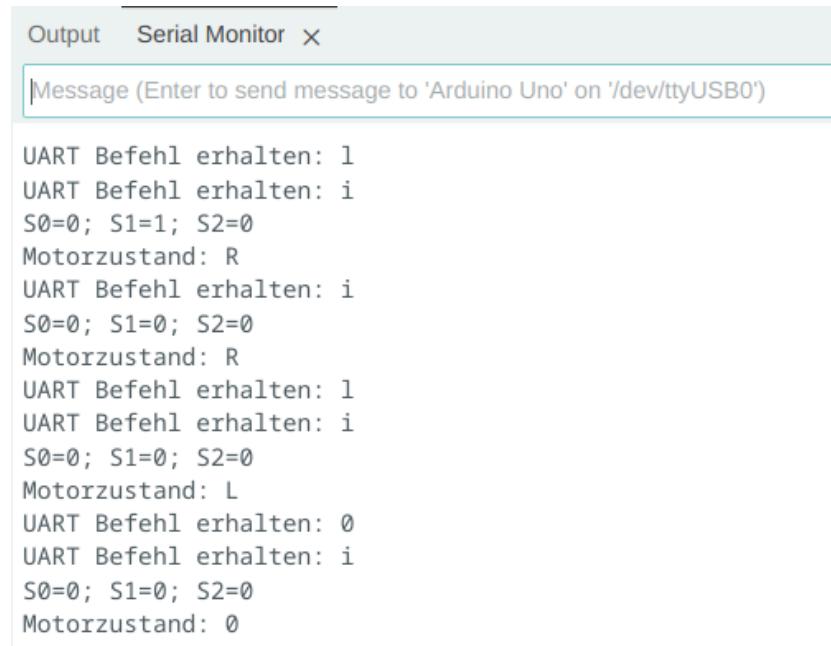
```

22
23     while (1) {
24         // Port Eingänge in Variablen speichern
25         // Taster S0 bis S2 (1=gedrückt)
26         const bool s0_aus = !((PINC >> PIN_S0_AUS) & 0x01);
27         const bool s1_rechts = !((PINC >> PIN_S1_RECHTS) & 0x01);
28         const bool s2_links = !((PINC >> PIN_S2_LINKS) & 0x01);
29
30         /*
31          * Motorsteuern über UART
32          * '0': ausschalten
33          * 'r': Rechtslauf
34          * 'l': Linkslauf
35          */
36         char uart_command = 0;
37
38         // Liegt eine Eingabe am UART vor?
39         if (kbhit()) {
40             uart_command = usart_getc_free();
41         }
42
43         if (uart_command == '0' || s0_aus) {
44             // Aus
45             q1_rechts = 0;
46             q2_links = 0;
47         } else if (uart_command == 'r' || s1_rechts) {
48             // Rechtsrum
49             q1_rechts = 1;
50             q2_links = 0;
51         } else if (uart_command == 'l' || s2_links) {
52             // Linksrum
53             q1_rechts = 0;
54             q2_links = 1;
55         }
56
57         // Taster Zustände senden
58         usart_puts("S0=");
59         usart_itoa(s0_aus);
60         usart_puts("; S1=");
61         usart_itoa(s1_rechts);
62         usart_puts("; S2=");
63         usart_itoa(s2_links);
64         usart_puts("\r\n");
65
66         if (q1_rechts) {
67             usart_puts("Motor dreht rechtsrum\r\n");
68         } else if (q2_links) {
69             usart_puts("Motor dreht linksrum\r\n");
70         }
71

```

```
72      // Alle Relais Pins (PC3 und PC4) werden geleert und dann gesetzt
73      PORTC = PORTC & ~(1 << PIN_Q1_RECHTS | 1 << PIN_Q2_LINKS) |
74          q1_rechts << PIN_Q1_RECHTS | q2_links << PIN_Q2_LINKS;
75    }
76 }
```

1.8 Aufgabe 11



The screenshot shows the Arduino Serial Monitor window. At the top, there are tabs for 'Output' and 'Serial Monitor' with a close button. Below the tabs, a message box contains the text: 'Message (Enter to send message to 'Arduino Uno' on '/dev/ttyUSB0')'. The main text area displays the following serial output:

```
UART Befehl erhalten: l
UART Befehl erhalten: i
S0=0; S1=1; S2=0
Motorzustand: R
UART Befehl erhalten: i
S0=0; S1=0; S2=0
Motorzustand: R
UART Befehl erhalten: l
UART Befehl erhalten: i
S0=0; S1=0; S2=0
Motorzustand: L
UART Befehl erhalten: 0
UART Befehl erhalten: i
S0=0; S1=0; S2=0
Motorzustand: 0
```

Abbildung 9: UART-Ausgabe von ET_50_KL_04

```
1 #define INFO "ET_50_KL_4\r\n"
2 #define BAUD 19200UL
3
4 #include "uartAT328p.h"
5 #include <stdbool.h>
6
7 // Relais Zustand
8 bool q1_rechts = 0;
9 bool q2_links = 0;
10
11 int main(void) {
12     // GPIO Ausgänge setzen
13     DDRB = 0x3F;
14     DDRC = 0x18;
15     DDRD = 0xFC;
16
17     // UART initialisieren
18     usart_init();
19
20     // Version über UART senden
21     usart_puts(INFO);
22
23     while (1) {
```

```

24 // Port Eingänge in Variablen speichern
25 // Taster S0 bis S2 (1=gedrückt)
26 const bool s0_aus = !((PINC >> PIN_S0_AUS) & 0x01);
27 const bool s1_rechts = !((PINC >> PIN_S1_RECHTS) & 0x01);
28 const bool s2_links = !((PINC >> PIN_S2_LINKS) & 0x01);
29
30 /*
31 * Motorsteuern über UART
32 * '0': ausschalten
33 * 'r': Rechtslauf
34 * 'l': Linkslauf
35 */
36 char uart_command = 0;
37
38 // Liegt eine Eingabe am UART vor?
39 if (kbhit()) {
40     uart_command = usart_getc_free();
41     usart_puts("UART Befehl erhalten: ");
42     usart_putc(uart_command);
43     usart_puts("\r\n");
44 }
45
46 if (uart_command == '0' || s0_aus) {
47     // Aus
48     q1_rechts = 0;
49     q2_links = 0;
50 } else if (uart_command == 'r' || s1_rechts) {
51     // Rechtsrum
52     q1_rechts = 1;
53     q2_links = 0;
54 } else if (uart_command == 'l' || s2_links) {
55     // Linksrsum
56     q1_rechts = 0;
57     q2_links = 1;
58 }
59
60 if (uart_command == 'i') {
61     // Taster Zustände senden
62     usart_puts("S0=");
63     usart_itoa(s0_aus);
64     usart_puts("; S1=");
65     usart_itoa(s1_rechts);
66     usart_puts("; S2=");
67     usart_itoa(s2_links);
68     usart_puts("\r\n");
69
70     usart_puts("Motorzustand: ");
71
72     if (q1_rechts) {
73         usart_putc('R');

```

```

74     } else if (q2_links) {
75         usart_putc('L');
76     } else {
77         usart_putc('0');
78     }
79
80     usart_puts("\r\n");
81 }
82
83 // Alle Relais Pins (PC3 und PC4) werden geleert und dann gesetzt
84 PORTC = PORTC & ~(1 << PIN_Q1_RECHTS | 1 << PIN_Q2_LINKS) |
85             q1_rechts << PIN_Q1_RECHTS | q2_links << PIN_Q2_LINKS;
86 }
87 }
```

1.9 Aufgabe 12

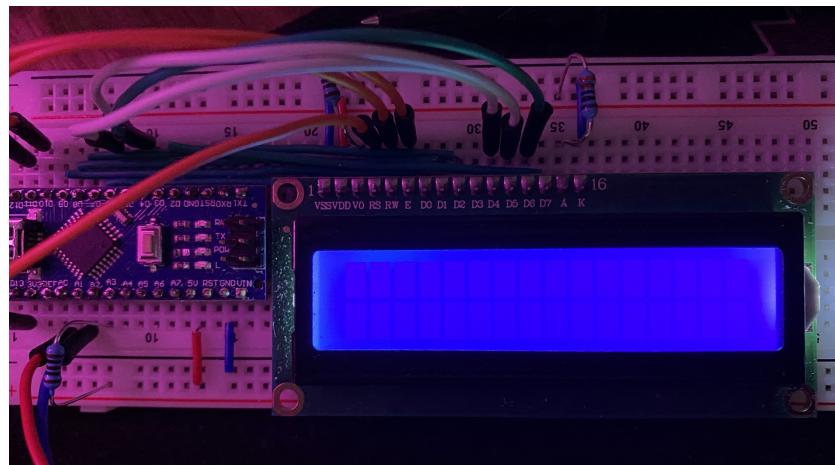


Abbildung 10: Initialisiertes LCD von ET_50_KL_05

```
1 #define INFO "ET_50_KL_5\r\n"
2 #define BAUD 19200UL
3 #define LCD_DELAY_MS 10
4
5 #include "LCD_HD44780.h"
6 #include "uartAT328p.h"
7 #include <stdbool.h>
8
9 // Relais Zustand
10 bool q1_rechts = 0;
11 bool q2_links = 0;
12
13 // Datenbits des Bus setzen. 0x11 => DB4=1; DBO=1
14 void set_data_bits(uint8_t data) {
15     // Alle betroffenen Pins zurücksetzen und dann setzen
16     // clang-format off
17     PORTB = PORTB & ~(1 << lcd_DB0 | 1 << lcd_DB1 | 1 << lcd_DB2)
18     | ((data >> 0) & 1) << lcd_DB0
19     | ((data >> 1) & 1) << lcd_DB1
20     | ((data >> 2) & 1) << lcd_DB2;
21
22     PORTD = PORTD & ~(1 << lcd_DB3 | 1 << lcd_DB4 | 1 << lcd_DB5
23     | 1 << lcd_DB6 | 1 << lcd_DB7)
24     | ((data >> 3) & 1) << lcd_DB3
25     | ((data >> 4) & 1) << lcd_DB4
26     | ((data >> 5) & 1) << lcd_DB5
27     | ((data >> 6) & 1) << lcd_DB6
28     | ((data >> 7) & 1) << lcd_DB7;
29 // clang-format on
```

```

30    }
31
32    // LCD Befehl senden
33    void command_lcd(uint8_t data) {
34        // Informationen bezüglich des HD44780 wurden dem Datenblatt entnommen:
35        // https://cdn.sparkfun.com/assets/9/5/f/7/b/HD44780.pdf
36
37        // Instructions (S. 24)
38        // Setze register select (RS) auf low zur Befehlannahme
39        PORTB &= ~(1 << lcd_RS | 1 << lcd_rw);
40
41        set_data_bits(data);
42        _delay_ms(LCD_DELAY_MS);
43
44        // Enable setzen
45        PORTB |= 1 << lcd_Enable;
46        _delay_ms(LCD_DELAY_MS);
47
48        // Enable ausschalten
49        PORTB &= ~(1 << lcd_Enable);
50    }
51
52    void initLCD() {
53        // Informationen bezüglich des HD44780 wurden dem Datenblatt entnommen:
54        // https://cdn.sparkfun.com/assets/9/5/f/7/b/HD44780.pdf
55
56        // Anleitung befindet sich unter "Initializing by Instruction" (S. 45)
57
58        // init delay after powerup
59        _delay_ms(50);
60
61        // Instruction Description (S. 25)
62        // Sets interface data length (DL = 1 for 8 bits),
63        // number of display lines (N = 1 for 2 lines),
64        // and character font (F = 0 for 5x8 dots).
65        const uint8_t function_set = 1 << 5 | 1 << 4 | 1 << 3;
66
67        // 3x Function Set
68        command_lcd(function_set);
69        _delay_ms(6);
70        command_lcd(function_set);
71        _delay_ms(1);
72        command_lcd(function_set);
73
74        // Setup
75        command_lcd(function_set);
76
77        // Sets entire display (D = 1) on/off,
78        // cursor on/off (C = 0),
79        // and blinking of cursor position character (B = 0).

```

```

80     command_lcd(1 << 3 | 1 << 2);
81
82     // Clear display
83     command_lcd(0x01);
84
85     // Entry mode
86     // Description: Sets cursor move direction and specifies display shift.
87     // These operations are performed during data write and read.
88
89     // Cursor automatisch weiterschieben: an
90     command_lcd(1 << 2 | 1 << 1);
91 }
92
93 int main(void) {
94     // GPIO Ausgänge setzen
95     DDRB = 0x3F;
96     DDRC = 0x18;
97     DDRD = 0xFC;
98
99     // UART initialisieren
100    usart_init();
101
102    // LCD initialisieren
103    initLCD();
104
105    // Version über UART senden
106    usart_puts(INFO);
107
108    while (1) {
109        // Port Eingänge in Variablen speichern
110        // Taster S0 bis S2 (1=gedrückt)
111        const bool s0_aus = !((PIN_C >> PIN_S0_AUS) & 0x01);
112        const bool s1_rechts = !((PIN_C >> PIN_S1_RECHTS) & 0x01);
113        const bool s2_links = !((PIN_C >> PIN_S2_LINKS) & 0x01);
114
115        /*
116         * Motorsteuern über UART
117         * '0': ausschalten
118         * 'r': Rechtslauf
119         * 'l': Linkslauf
120         */
121        char uart_command = 0;
122
123        // Liegt eine Eingabe am UART vor?
124        if (kbhit()) {
125            uart_command = usart_getc_free();
126            usart_puts("UART Befehl erhalten: ");
127            usart_putc(uart_command);
128            usart_puts("\r\n");
129        }

```

```

130
131     if (uart_command == '0' || s0_aus) {
132         // Aus
133         q1_rechts = 0;
134         q2_links = 0;
135     } else if (uart_command == 'r' || s1_rechts) {
136         // Rechtsrum
137         q1_rechts = 1;
138         q2_links = 0;
139     } else if (uart_command == 'l' || s2_links) {
140         // Linksrum
141         q1_rechts = 0;
142         q2_links = 1;
143     }
144
145     if (uart_command == 'i') {
146         // Taster Zustände senden
147         usart_puts("S0=");
148         usart_itoa(s0_aus);
149         usart_puts("; S1=");
150         usart_itoa(s1_rechts);
151         usart_puts("; S2=");
152         usart_itoa(s2_links);
153         usart_puts("\r\n");
154
155         usart_puts("Motorzustand: ");
156
157         if (q1_rechts) {
158             usart_putc('R');
159         } else if (q2_links) {
160             usart_putc('L');
161         } else {
162             usart_putc('0');
163         }
164
165         usart_puts("\r\n");
166     }
167
168     // Alle Relais Pins (PC3 und PC4) werden geleert und dann gesetzt
169     PORTC = PORTC & ~(1 << PIN_Q1_RECHTS | 1 << PIN_Q2_LINKS) |
170             q1_rechts << PIN_Q1_RECHTS | q2_links << PIN_Q2_LINKS;
171 }
172 }
```

1.10 Aufgabe 13



Abbildung 11: Version und Motorzustand auf LCD von ET_50_KL_06

```
1 #define INFO "ET_50_KL_6"
2 #define BAUD 19200UL
3 #define LCD_DELAY_MS 10
4
5 #include "LCD_HD44780.h"
6 #include "uartAT328p.h"
7 #include <stdbool.h>
8
9 // Relais Zustand
10 bool q1_rechts = 0;
11 bool q2_links = 0;
12
13 // Datenbits des Bus setzen. 0x11 => DB4=1; DBO=1
14 void set_data_bits(uint8_t data) {
15     // Alle betroffenen Pins zurücksetzen und dann setzen
16     // clang-format off
17     PORTB = PORTB & ~(1 << lcd_DB0 | 1 << lcd_DB1 | 1 << lcd_DB2)
18     | ((data >> 0) & 1) << lcd_DB0
19     | ((data >> 1) & 1) << lcd_DB1
20     | ((data >> 2) & 1) << lcd_DB2;
21
22     PORTD = PORTD & ~(1 << lcd_DB3 | 1 << lcd_DB4 | 1 << lcd_DB5
23     | 1 << lcd_DB6 | 1 << lcd_DB7)
24     | ((data >> 3) & 1) << lcd_DB3
25     | ((data >> 4) & 1) << lcd_DB4
26     | ((data >> 5) & 1) << lcd_DB5
27     | ((data >> 6) & 1) << lcd_DB6
28     | ((data >> 7) & 1) << lcd_DB7;
29     // clang-format on
30 }
31
32 // LCD Befehl senden
```

```

33 void command_lcd(uint8_t data) {
34     // Informationen bezüglich des HD44780 wurden dem Datenblatt entnommen:
35     // https://cdn.sparkfun.com/assets/9/5/f/7/b/HD44780.pdf
36
37     // Instructions (S. 24)
38     // Setze register select (RS) auf low zur Befehlannahme
39     PORTB &= ~(1 << lcd_RS | 1 << lcd_rw);
40
41     set_data_bits(data);
42     _delay_ms(LCD_DELAY_MS);
43
44     // Enable setzen
45     PORTB |= 1 << lcd_Enable;
46     _delay_ms(LCD_DELAY_MS);
47
48     // Enable ausschalten
49     PORTB &= ~(1 << lcd_Enable);
50 }
51
52 void initLCD() {
53     // Informationen bezüglich des HD44780 wurden dem Datenblatt entnommen:
54     // https://cdn.sparkfun.com/assets/9/5/f/7/b/HD44780.pdf
55
56     // Anleitung befindet sich unter "Initializing by Instruction" (S. 45)
57
58     // init delay after powerup
59     _delay_ms(50);
60
61     // Instruction Description (S. 25)
62     // Sets interface data length (DL = 1 for 8 bits),
63     // number of display lines (N = 1 for 2 lines),
64     // and character font (F = 0 for 5x8 dots).
65     const uint8_t function_set = 1 << 5 | 1 << 4 | 1 << 3;
66
67     // 3x Function Set
68     command_lcd(function_set);
69     _delay_ms(6);
70     command_lcd(function_set);
71     _delay_ms(1);
72     command_lcd(function_set);
73
74     // Setup
75     command_lcd(function_set);
76
77     // Sets entire display (D = 1) on/off,
78     // cursor on/off (C = 0),
79     // and blinking of cursor position character (B = 0).
80     command_lcd(1 << 3 | 1 << 2);
81
82     // Clear display

```

```

83     command_lcd(0x01);
84
85     // Entry mode
86     // Description: Sets cursor move direction and specifies display shift.
87     // These operations are performed during data write and read.
88
89     // Cursor automatisch weiterschieben: an
90     command_lcd(1 << 2 | 1 << 1);
91 }
92
93 int main(void) {
94     // GPIO Ausgänge setzen
95     DDRB = 0x3F;
96     DDRC = 0x18;
97     DDRD = 0xFC;
98
99     // UART initialisieren
100    usart_init();
101
102    // Version über UART senden
103    usart_puts(INFO);
104    usart_puts("\r\n");
105
106    // LCD initialisieren
107    initLCD();
108
109    // Version anzeigen
110    lcd_puts(INFO);
111
112    while (1) {
113        // Port Eingänge in Variablen speichern
114        // Taster S0 bis S2 (1=gedrückt)
115        const bool s0_aus = !((PINC >> PIN_S0_AUS) & 0x01);
116        const bool s1_rechts = !((PINC >> PIN_S1_RECHTS) & 0x01);
117        const bool s2_links = !((PINC >> PIN_S2_LINKS) & 0x01);
118
119        /*
120         * Motorsteuern über UART
121         * '0': ausschalten
122         * 'r': Rechtslauf
123         * 'l': Linkslauf
124         */
125        char uart_command = 0;
126
127        // Liegt eine Eingabe am UART vor?
128        if (kbhit()) {
129            uart_command = usart_getc_free();
130            usart_puts("UART Befehl erhalten: ");
131            usart_putc(uart_command);
132            usart_puts("\r\n");

```

```

133     }
134
135     if (uart_command == '0' || s0_aus) {
136         // Aus
137         q1_rechts = 0;
138         q2_links = 0;
139     } else if (uart_command == 'r' || s1_rechts) {
140         // Rechtsrum
141         q1_rechts = 1;
142         q2_links = 0;
143     } else if (uart_command == 'l' || s2_links) {
144         // Linksrum
145         q1_rechts = 0;
146         q2_links = 1;
147     }
148
149     if (q1_rechts) {
150         lcd_gotoxy(0, 1);
151         lcd_puts("Rechtslauf      ");
152     } else if (q2_links) {
153         lcd_gotoxy(0, 1);
154         lcd_puts("Linkslauf      ");
155     } else {
156         lcd_gotoxy(0, 1);
157         lcd_puts("Ausgeschalten   ");
158     }
159
160     if (uart_command == 'i') {
161         // Taster Zustände senden
162         usart_puts("S0=");
163         usart_itoa(s0_aus);
164         usart_puts("; S1=");
165         usart_itoa(s1_rechts);
166         usart_puts("; S2=");
167         usart_itoa(s2_links);
168         usart_puts("\r\n");
169
170         usart_puts("Motorzustand: ");
171
172         if (q1_rechts) {
173             usart_putc('R');
174         } else if (q2_links) {
175             usart_putc('L');
176         } else {
177             usart_putc('0');
178         }
179
180         usart_puts("\r\n");
181     }
182

```

```
183     // Alle Relais Pins (PC3 und PC4) werden geleert und dann gesetzt
184     PORTC = PORTC & ~(1 << PIN_Q1_RECHTS | 1 << PIN_Q2_LINKS) |
185             q1_rechts << PIN_Q1_RECHTS | q2_links << PIN_Q2_LINKS;
186 }
187 }
```

1.11 Aufgabe 14

```
1  /*
2   * Warnung: Boolische True-Werte müssen den Wert 1 haben
3   *
4   * Zuweisungsliste:
5   *
6   * Arduino | MCU Pin | Bauteil Pin | Bezeichnung
7   * 5V      Vcc
8   * GND    Vss
9   * 13     PB5      RS      LCD
10  * 12    PB4      R/~W    LCD
11  * 11    PB3      E       LCD
12  * 10    PB2      D0     8 bit databus
13  * 9     PB1      D1     8 bit databus
14  * 8     PB0      D2     8 bit databus
15  * 7     PD7      D3     8 bit databus
16  * 6     PD6      D4     8 bit databus
17  * 5     PD5      D5     8 bit databus
18  * 4     PD4      D6     8 bit databus
19  * 3     PD3      D7     8 bit databus
20  * 2     PD2      CLK    Datenübernehmen für 7 segment display
21  * A0    PC0      S0    Taster in Negativlogik
22  * A1    PC1      S1    Taster in Negativlogik
23  * A2    PC2      S2    Taster in Negativlogik
24  * A3    PC3      Q1    Relais Rechts
25  * A4    PC4      Q2    Relais Links
26  * A5    PC5
27  */
28
29 #define PIN_S0_AUS PC0
30 #define PIN_S1_RECHTS PC1
31 #define PIN_S2_LINKS PC2
32 #define PIN_Q1_RECHTS PC3
33 #define PIN_Q2_LINKS PC4
34
35 #define F_CPU 16000000
36
37 // Define constants
38 #define INFO "ET_50_KL_7"
39 #define BAUD 19200UL
40 #define LCD_DELAY_MS 1
41
42 #include "LCD_HD44780.h"
43 #include "uartAT328p.h"
44 #include <stdbool.h>
45
46 // Relais Zustand
47 bool q1_rechts = 0;
48 bool q2_links = 0;
49
50 // Datenbits des Bus setzen. 0x11 => DB4=1; DB0=1
```

```

14 void set_data_bits(uint8_t data) {
15     // Alle betroffenen Pins zurücksetzen und dann setzen
16     // clang-format off
17     PORTB = PORTB & ~(1 << lcd_DB0 | 1 << lcd_DB1 | 1 << lcd_DB2)
18     | ((data >> 0) & 1) << lcd_DB0
19     | ((data >> 1) & 1) << lcd_DB1
20     | ((data >> 2) & 1) << lcd_DB2;
21
22     PORTD = PORTD & ~(1 << lcd_DB3 | 1 << lcd_DB4 | 1 << lcd_DB5
23     | 1 << lcd_DB6 | 1 << lcd_DB7)
24     | ((data >> 3) & 1) << lcd_DB3
25     | ((data >> 4) & 1) << lcd_DB4
26     | ((data >> 5) & 1) << lcd_DB5
27     | ((data >> 6) & 1) << lcd_DB6
28     | ((data >> 7) & 1) << lcd_DB7;
29     // clang-format on
30 }
31
32 // LCD Befehl senden
33 void command_lcd(uint8_t data) {
34     // Informationen bezüglich des HD44780 wurden dem Datenblatt entnommen:
35     // https://cdn.sparkfun.com/assets/9/5/f/7/b/HD44780.pdf
36
37     // Instructions (S. 24)
38     // Setze register select (RS) auf low zur Befehlannahme
39     PORTB &= ~(1 << lcd_RS | 1 << lcd_rw);
40
41     set_data_bits(data);
42     _delay_ms(LCD_DELAY_MS);
43
44     // Enable setzen
45     PORTB |= 1 << lcd_Enable;
46     _delay_ms(LCD_DELAY_MS);
47
48     // Enable ausschalten
49     PORTB &= ~(1 << lcd_Enable);
50 }
51
52 void initLCD() {
53     // Informationen bezüglich des HD44780 wurden dem Datenblatt entnommen:
54     // https://cdn.sparkfun.com/assets/9/5/f/7/b/HD44780.pdf
55
56     // Anleitung befindet sich unter "Initializing by Instruction" (S. 45)
57
58     // init delay after powerup
59     _delay_ms(50);
60
61     // Instruction Description (S. 25)
62     // Sets interface data length (DL = 1 for 8 bits),
63     // number of display lines (N = 1 for 2 lines),

```

```

64 // and character font (F = 0 for 5x8 dots).
65 const uint8_t function_set = 1 << 5 | 1 << 4 | 1 << 3;
66
67 // 3x Function Set
68 command_lcd(function_set);
69 _delay_ms(6);
70 command_lcd(function_set);
71 _delay_ms(1);
72 command_lcd(function_set);
73
74 // Setup
75 command_lcd(function_set);
76
77 // Sets entire display (D = 1) on/off,
78 // cursor on/off (C = 0),
79 // and blinking of cursor position character (B = 0).
80 command_lcd(1 << 3 | 1 << 2);
81
82 // Clear display
83 command_lcd(0x01);
84
85 // Entry mode
86 // Description: Sets cursor move direction and specifies display shift.
87 // These operations are performed during data write and read.
88
89 // Cursor automatisch weiterschieben: an
90 command_lcd(1 << 2 | 1 << 1);
91 }
92
93 // Zeigt Zahl auf 7 Segment-Anzeige über 74HC573 (Register)
94 // @param led_states Binary layout: Ob{a,b,c,d,e,f,g,dp}
95 void set_7_segment_display(uint8_t led_states) {
96     set_data_bits(led_states);
97     _delay_ms(LCD_DELAY_MS);
98
99     // Enable setzen
100    PORTD |= 1 << PD2;
101    _delay_ms(LCD_DELAY_MS);
102
103    // Enable ausschalten
104    PORTD &= ~(1 << PD2);
105 }
106
107 int main(void) {
108     // GPIO Ausgänge setzen
109     DDRB = 0x3F;
110     DDRC = 0x18;
111     DDRD = 0xFC;
112
113     // UART initialisieren

```

```

114     usart_init();
115
116     // Version über UART senden
117     usart_puts(INFO);
118     usart_puts("\r\n");
119
120     // LCD initialisieren
121     initLCD();
122
123     // Version anzeigen
124     lcd_puts(INFO);
125
126     while (1) {
127         // Port Eingänge in Variablen speichern
128         // Taster S0 bis S2 (1=gedrückt)
129         const bool s0_aus = !((PINC >> PIN_S0_AUS) & 0x01);
130         const bool s1_rechts = !((PINC >> PIN_S1_RECHTS) & 0x01);
131         const bool s2_links = !((PINC >> PIN_S2_LINKS) & 0x01);
132
133         /*
134          * Motorsteuern über UART
135          * '0': ausschalten
136          * 'r': Rechtslauf
137          * 'l': Linkslauf
138          */
139         char uart_command = 0;
140
141         // Liegt eine Eingabe am UART vor?
142         if (kbhit()) {
143             uart_command = usart_getc_free();
144             usart_puts("UART Befehl erhalten: ");
145             usart_putc(uart_command);
146             usart_puts("\r\n");
147         }
148
149         if (uart_command == '0' || s0_aus) {
150             // Aus
151             q1_rechts = 0;
152             q2_links = 0;
153         } else if (uart_command == 'r' || s1_rechts) {
154             // Rechtsrum
155             q1_rechts = 1;
156             q2_links = 0;
157         } else if (uart_command == 'l' || s2_links) {
158             // Linksrumb
159             q1_rechts = 0;
160             q2_links = 1;
161         }
162
163         if (q1_rechts) {

```

```

164     lcd_gotoxy(0, 1);
165     lcd_puts("Rechtslauf      ");
166
167     // 'r' => 0000 1010 => 0x0A
168     set_7_segment_display(0x3A);
169 } else if (q2_links) {
170     lcd_gotoxy(0, 1);
171     lcd_puts("Linkslauf      ");
172
173     // 'l' => 0001 1100 => 0x1C
174     set_7_segment_display(0x1C);
175 } else {
176     lcd_gotoxy(0, 1);
177     lcd_puts("Ausgeschalten   ");
178
179     // 'o' => 0011 1010 => 0x3A
180     set_7_segment_display(0x3A);
181 }
182
183 if (uart_command == 'i') {
184     // Taster Zustände senden
185     usart_puts("S0=");
186     usart_itoa(s0_aus);
187     usart_puts("; S1=");
188     usart_itoa(s1_rechts);
189     usart_puts("; S2=");
190     usart_itoa(s2_links);
191     usart_puts("\r\n");
192
193     usart_puts("Motorzustand: ");
194
195     if (q1_rechts) {
196         usart_putc('R');
197     } else if (q2_links) {
198         usart_putc('L');
199     } else {
200         usart_putc('0');
201     }
202
203     usart_puts("\r\n");
204 }
205
206 // Alle Relais Pins (PC3 und PC4) werden geleert und dann gesetzt
207 PORTC = PORTC & ~(1 << PIN_Q1_RECHTS | 1 << PIN_Q2_LINKS) |
208     q1_rechts << PIN_Q1_RECHTS | q2_links << PIN_Q2_LINKS;
209 }
210 }
```

1.12 Aufgabe 15

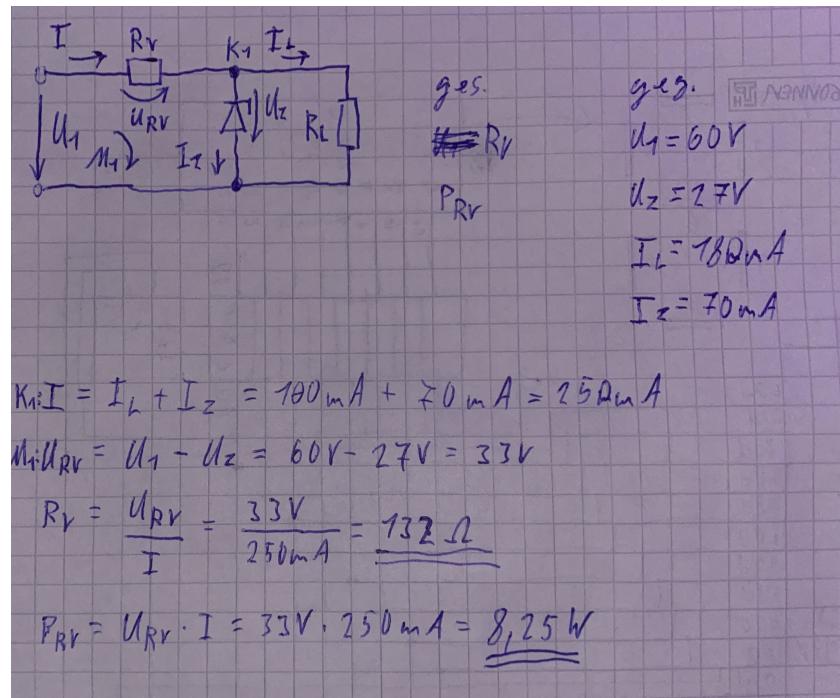


Abbildung 12: FRB Kapitel 6.2.3.1

1.13 Aufgabe 16

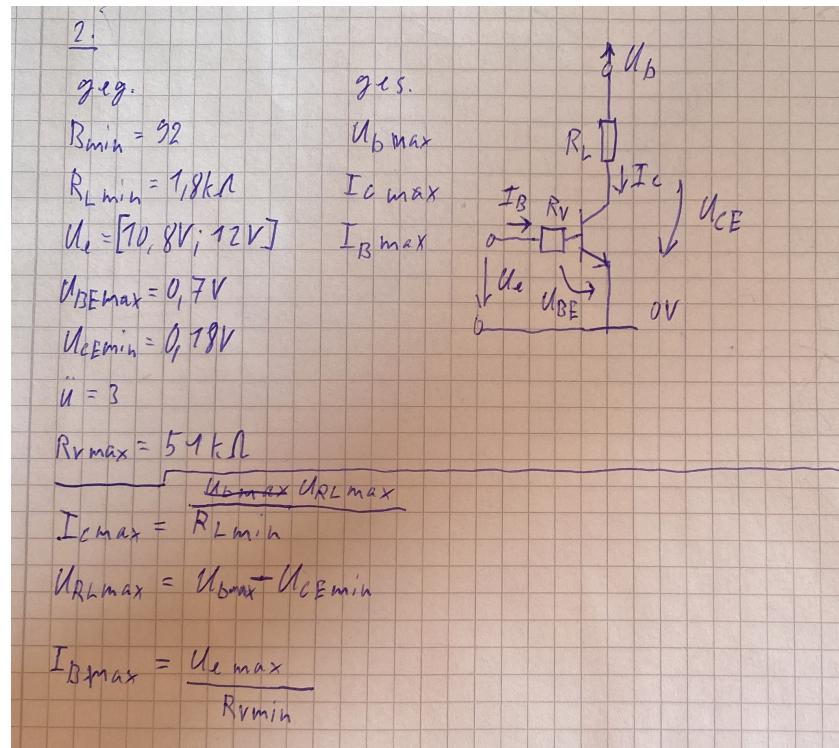


Abbildung 13: FRB Kapitel 6.10.1 Nr. 2

Ich kam bei dieser Aufgabe nicht auf einen Lösungsweg.

1.14 Aufgabe 17

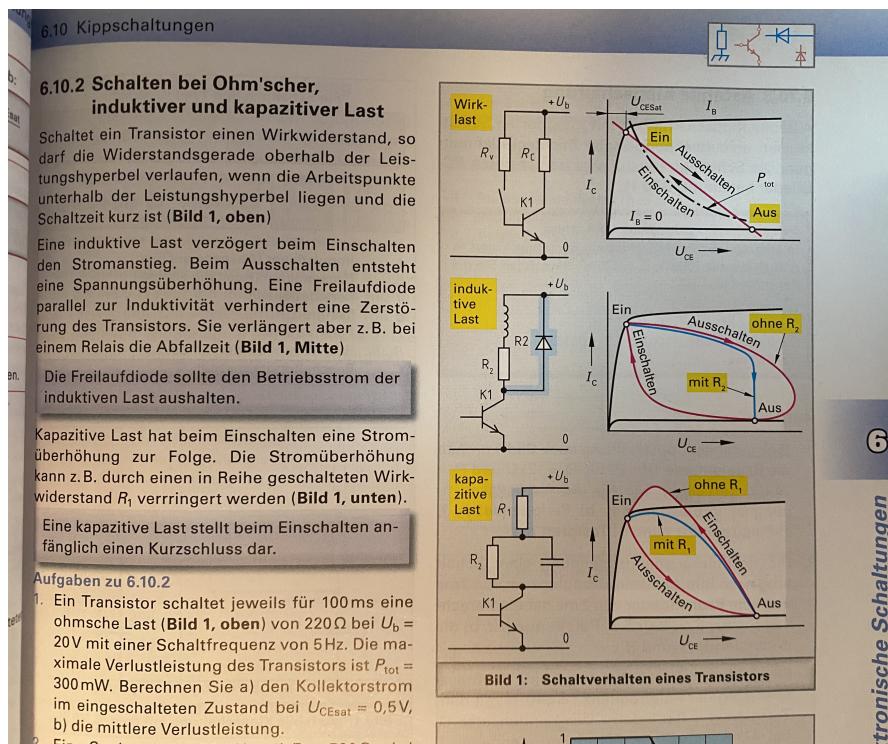


Abbildung 14: FRB Kapitel 6.10.2 Buchseite

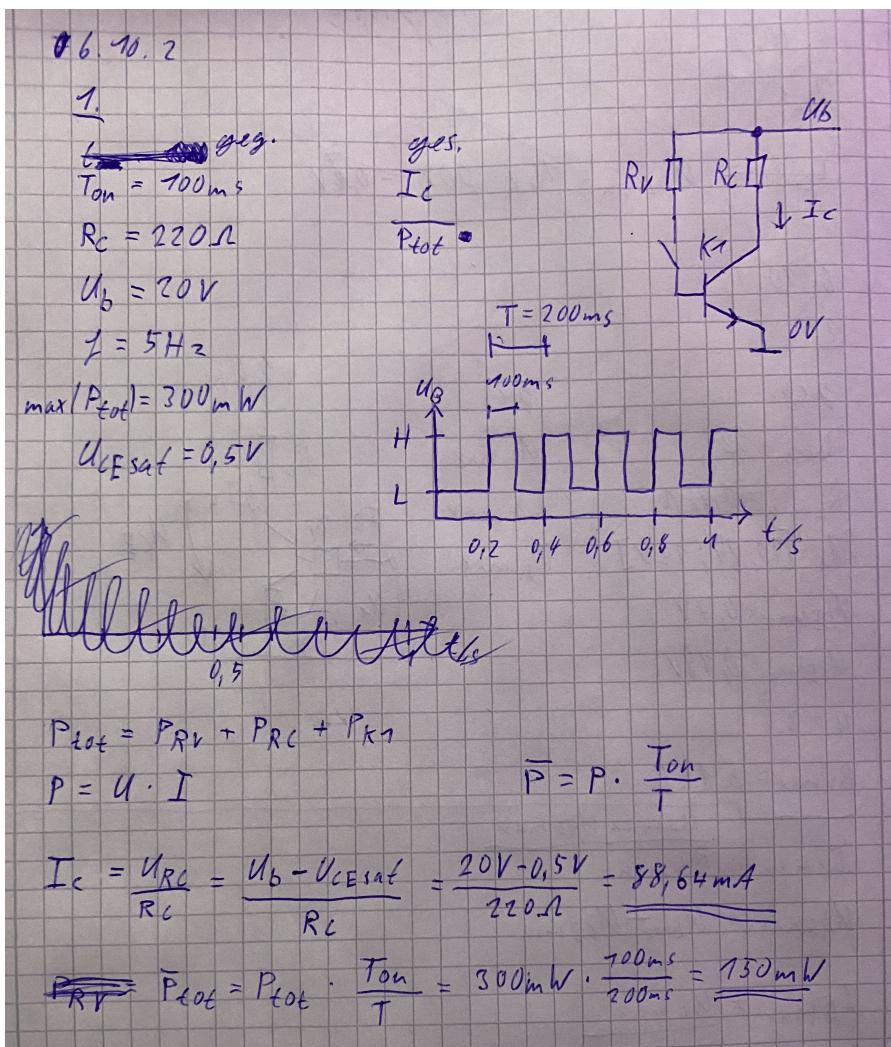


Abbildung 15: FRB Kapitel 6.10.2 Nr. 1

Nr	Fachkompetenz: Ich kann,	Tax moodle	Tätigkeitsbeschreibung: Ich habe,	T N W
0	für einen DC Motor (Lüfter) 12V / 600 W ein Rechts-/Linkslaufsteuerung mit Relais und galvanisch getrennt skizziert. VCC1	☺ ☺ 0,5	die Schaltungsvorlage mit allen benötigten Bauteilen und Verbindungen ergänzt.	
1	für ein Relais 5V / 2,5W eine Treiberstufe mit einem NPN Transistoren für einen TTL MC-PORT skizziert. VCC2	☺ ☺ 0,5	die Schaltungsvorlage mit allen benötigten Bauteilen und Verbindungen ergänzt.	
2	die Aufgabe 0 und 1 erklären.	☺ ☺ 0,5	die Schaltungen erklärt.	
3	die Aufgabe 1 <u>dimensionieren</u> .	☺ ☺ ☺ 3,0	die Schaltung für einen BC337/40 mit $U_{BE}=0,85V$, $B=350$, $U_{CESAT}=0,7V$, MC-PORT $U_{Hmin}=2,4V$, $I_{Hmax}=10mA$ dimensioniert.	
4	die Schaltungsvorlage mit einer kompletten MC-Basis-Schaltung (Quarz, Reset, ADC [AREF=VCC2], RS232/USB und VCC2)	☺ ☺ 2,0	die Schaltungsvorlage mit allen notwendigen Bauelementen + Werten inkl. Spannungszufuhr VCC2 USB → 5V ergänzt.	
5	3 Taster an einen MC in Negativlogik anschließen.	☺ ☺ 1,5	die Schaltungsvorlage aus dem Unterricht ergänzt und erklärt. (S0 = Aus, S1 = R, S2 = L)	
6	den 8 Bit Modus des HD44780 16x2 erklären.	☺ ☺ 2,5	die Schaltungsvorlage aus dem Unterricht ergänzt und erklärt.	
7	den 8 Bit Datenbus des HD44780, um eine 7 Seg. Anzeige mit 74HC573 ergänzen.	☺ ☺ 2,5	die Schaltungsvorlage analog zum Unterricht mit allen Bauteilen + Werten ergänzt und erklärt.	
8	ein MC Programm schreiben, das den Zustand der Taster aus A5 abfragt und per UART sendet. Grundsätzlich ist beim Programmstart die aktuelle Versionsnummer auszugeben.	☺ ☺ 2,0	das Basisprogramm ET_50_KL_01 im AVR-Studio (in C) mit dieser Funktion erstellt. BAUD-Rate = 19200 (ZWL und selbsterklärende Kommentare, Modularer Aufbau für alle Programme!!)	
9	das MC Programm aus A8 als Basis mit folgender Funktion ergänzen: Mit S1 wird das Relais R angesteuert, mit S2 wird das Relais L angesteuert, mit S0 werden alle Relais ausgeschaltet. Die Zustände der Relais werden gehalten und per UART gesendet.	☺ ☺ 2,0	das Basisprogramm ET_50_KL_01 zu ET_50_KL_02 kopiert und im AVR-Studio mit dieser Funktion ergänzt. Ausgaben sind mit Screenshots zu belegen.	
10	das Programm aus A9, um folgende Funktion ergänzen: empfängt das MC-Programm ein '0' → Motor ausschalten '1' → Motor Linkslauf einschalten 'r' → Motor Rechtslauf einschalten	☺ ☺ 2,0	das Basisprogramm ET_50_KL_02 zu ET_50_KL_03 kopiert und im AVR-Studio mit dieser Funktion ergänzt.	
11	das Programm aus A9, um folgende Funktion ergänzen: Der Zustand der Relais aus A10 auf Anfrage per RS232/USB mit einem 'i' beantwortet. ('L' → Links ; 'R' → Rechts ; '0' → aus)	☺ ☺ 2,0	das Basisprogramm ET_50_KL_03 zu ET_50_KL_04 kopiert und im AVR-Studio mit dieser Funktion ergänzt.	
12	die Initialisierung des LCD in A11 einbinden initLCD() und ausführlich erklären.	☺ ☺ 2,0	das Basisprogramm ET_50_KL_04 zu ET_50_KL_05 kopiert und im AVR-Studio mit dieser Funktion ergänzt.	
13	das Programm aus A12, um folgende Funktion ergänzen: Die Versionsnummer und der Motorzustand wird auf dem LCD	☺ ☺ 2,0	das Basisprogramm ET_50_KL_05 zu ET_50_KL_06 kopiert und im AVR-Studio mit dieser Funktion ergänzt.	

ET_50_KL_Maschinen_Transistor_Digitaltechnik_MC

	angezeigt.			
14	das Programm aus A13, um folgende Funktion ergänzen: Der Motorzustand wird auf der LED-Anzeige wie folgt dargestellt. 'o' = aus, 'l' = Linkslauf, 'r'=Rechtslauf	☺ ☺ 2,0	das Basisprogramm ET_50_KL_06 zu ET_50_KL_07 kopiert und im AVR-Studio mit dieser Funktion ergänzt.	
15	die Aufgabe 6.2.3.1 Nr. 3 lösen.	☺ ☺ 0,5	die Aufgabe ausführlich gelöst.	
16	die Aufgabe 6.10.1 Nr. 2, 5, 6 lösen.	☺ ☺ 1,5	die Aufgaben ausführlich gelöst.	
17	die Aufgaben 6.10.2 Nr. 1,2,3,4 lösen.	☺ ☺ 2,0	die Aufgaben ausführlich gelöst.	
19	die Aufgaben 6.10.3 Nr. 1,2,3,4,6 lösen.	☺ ☺ 2,5	die Aufgaben ausführlich gelöst.	
20	die Aufgaben 6.10.4 Nr. 1,3,4 lösen.	☺ ☺ 1,5	die Aufgaben ausführlich gelöst.	
21	die Aufgaben 6.10.5 Nr. 1,2,3 lösen.	☺ ☺ 1,5	die Aufgaben ausführlich gelöst.	
22	die Aufgaben 7.5.3 Nr. 1,2,3,4,7 lösen.	☺ ☺ 2,5	die Aufgaben ausführlich gelöst.	
23	die Aufgaben 7.5.4 Nr. 1,2,3,4 lösen.	☺ ☺ 2,0	die Aufgaben ausführlich gelöst.	
24	die Aufgaben 7.6.4 Nr. 1,2,3,4 lösen.	☺ ☺ 2,0	die Aufgaben ausführlich gelöst.	
25	die Aufgaben 7.7 Nr. 1,2,3,4 lösen.	☺ ☺ 2,0	die Aufgaben ausführlich gelöst.	
26	die Aufgaben 8.2 Nr. 1,2,3,4,6,7,8 lösen.	☺ ☺ 3,0	die Aufgaben ausführlich gelöst.	
27	die Aufgaben 8.3 Nr. 3 lösen.	☺ ☺ 1,0	die Aufgaben ausführlich gelöst.	
28	die Aufgaben 8.5 Nr. 1,5 lösen.	☺ ☺ 2,0	die Aufgaben ausführlich gelöst.	
29	die Aufgaben 8.6 Nr. 1,3 lösen.	☺ ☺ 2,0	die Aufgaben ausführlich gelöst.	

**Alle vorherigen Kann-Listen gelten als Grundlagen
und**

sind ebenfalls prüfungsrelevant.

**Es ist immer der vollständige Lösungsweg
inkl. Betriebsmittelkennzeichnung usw. anzugeben !!!**

Für die Bearbeitung sind das **Datenblatt ATMEGA328p.pdf** und **Mikrocontroller.net** wichtig.

<https://www.mikrocontroller.net/articles/AVR-Tutorial>

<https://www.mikrocontroller.net/articles/AVR-GCC-Tutorial>

Alle vorherigen Kann-Listen sind ebenfalls Gegenstand der Klausur.

Unterrichtsmitschriften, FKB, FRB und Projektvorträge, Videos zum Thema

<https://www.youtube.com/playlist?list=PLBF281451AA9A2E13> , <https://www.elektroniktutor.de> u.a.

☺ Reproduktion	☺ ☺ Reorganisation	☺ ☺ ☺ Anwendung	☺ ☺ ☺ ☺ Problemlösung
----------------	--------------------	-----------------	-----------------------

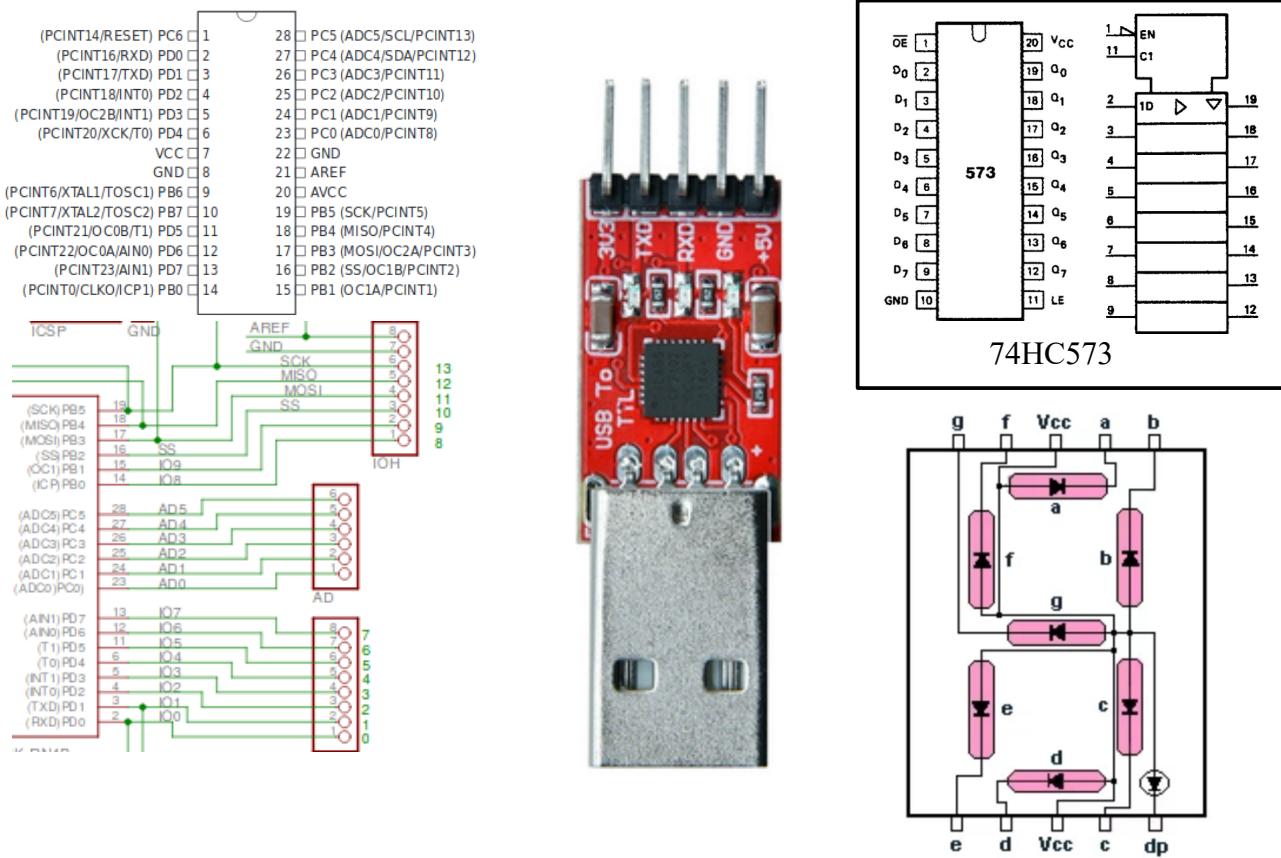
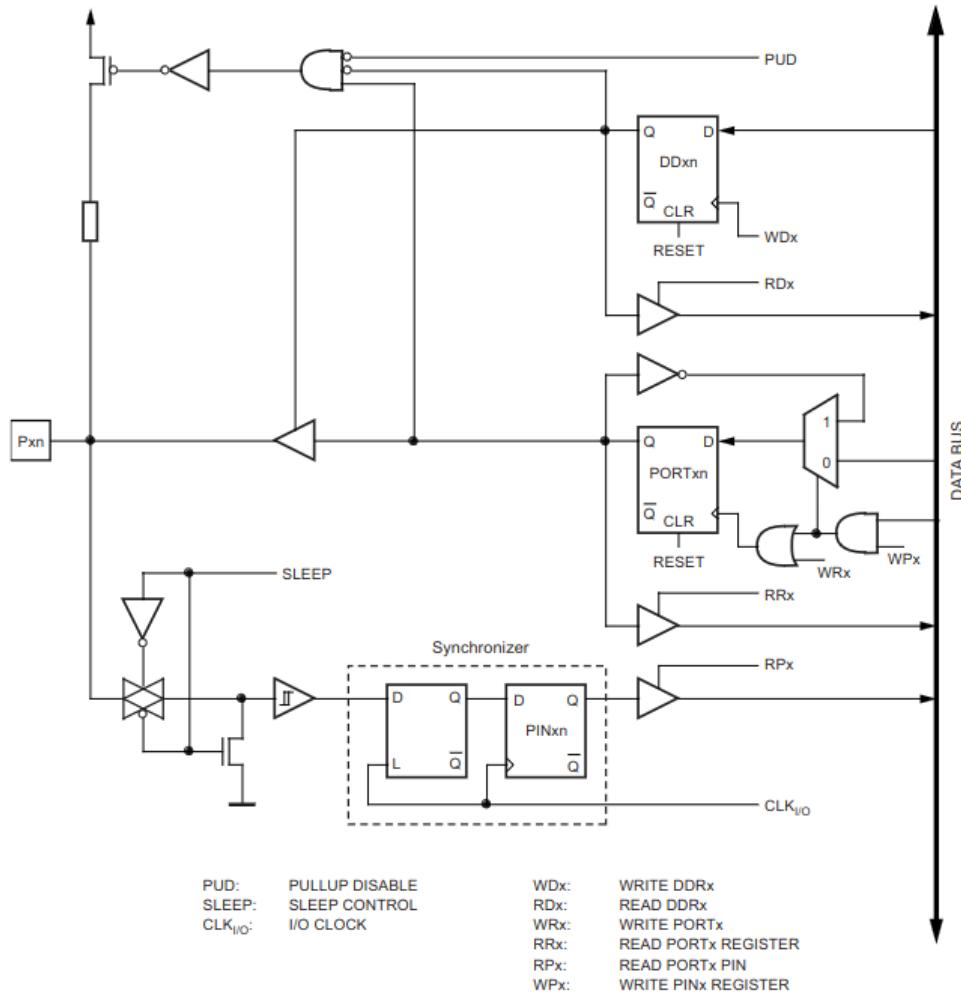
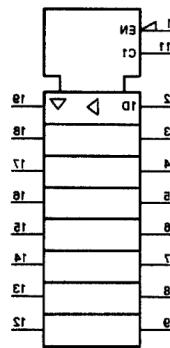
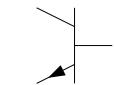
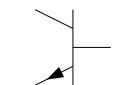
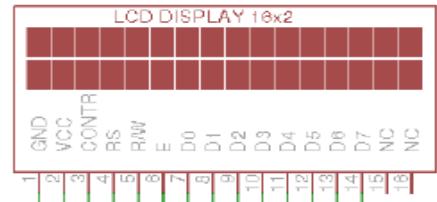
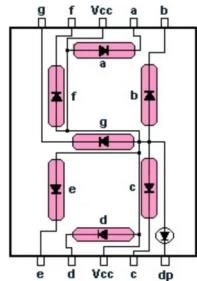


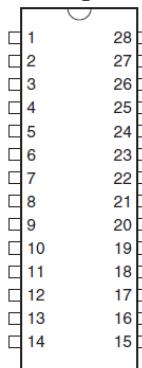
Figure 13-2. General Digital I/O⁽¹⁾



VCC 2
+5V /USB



Atmega328



Motor ZL

VCC 1
+12V
GND 1



GND 2
USB

