

# Praktikum Mikrocomputertechnik (EIT - B3, MEC - B5)

## Versuch 02 - Schrittmotoransteuerung

Version: 2024-10-18a

### Inhaltsverzeichnis

1	Aufgabenstellung .....	2
2	Informationssammlung .....	2
2.1	Allgemeines .....	2
2.2	Treiberplatine .....	2
2.3	Schrittmotor .....	3
2.4	Anschlußbelegung .....	4
3	Aufgabe 01: Lauflicht für Anfänger .....	5
4	Aufgabe 02: Lauflicht für fortgeschrittene Programmierer .....	6
4.1	Aufgabenstellung .....	6
5	Aufgabe 03: Lauflicht ändert Richtung auf Tastendruck .....	7
5.1	Aufgabenstellung .....	7
5.2	Musterlösung .....	7
6	Aufgabe 04: Ansteuerung eines Schrittmotors im Vollschrittbetrieb und im Halbschrittbetrieb .....	8
6.1	Aufgabenstellung .....	8
6.2	Informationssammlung .....	8
6.3	Umsetzung .....	9

## 1 Aufgabenstellung

In diesem Praktikum wollen wir zunächst vier LEDs und anschließend einen Schrittmotor mit digitalen Ausgängen des MSP430 ansteuern.

Durch Drücken des Tasters S2 soll der Motor seine Drehrichtung ändern.

## 2 Informationssammlung

### 2.1 Allgemeines

In diesem Praktikum wollen wir zunächst 4 LEDs und im weiteren Verlauf einen Schrittmotor ansteuern.

Die LEDs befinden sich auf der in [2.2](#) beschriebenen Platine. Diese Platine wird auch als Treiberplatine für den Schrittmotor (siehe [2.3](#)) verwendet.

Um den Schrittmotor zu bewegen, müssen seine Spulen mit einem bestimmten Muster angesteuert werden. Siehe hierzu [6.2](#).

### 2.2 Treiberplatine

ULN2003 Four-phase Five-wire Driver Board



**Features:**

A, B, C, D LEDs indicate the shape of the four-phase stepper motor

It is equipped with a standard interface for stepper motors and can be inserted directly during use.


Dimensions: 31 × 35mm

<https://learning-campus.th-rosenheim.de/course/view.php?id=9266#>


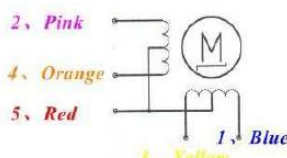
## 2.3 Schrittmotor

**28BYJ-48 – 5V Stepper Motor**

The 28BYJ-48 is a small stepper motor suitable for a large range of applications.



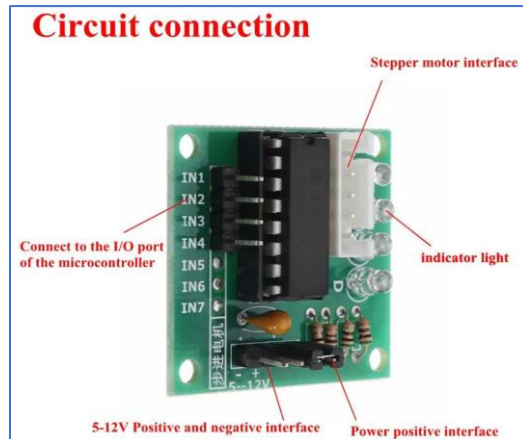
Rated voltage :	5VDC
Number of Phase	4
Speed Variation Ratio	1/64
Stride Angle	5.625°/64
Frequency	100Hz
DC resistance	50Ω±7%(25°C)
Idle In-traction Frequency	> 600Hz
Idle Out-traction Frequency	> 1000Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm
Insulated resistance	>10MQ(500V)
Insulated electricity power	600VAC/1mA/1s
Insulation grade	A
Rise in Temperature	<40K(120Hz)
Noise	<35dB(120Hz, No load, 10cm)
Model	28BYJ-48 – 5V

<https://learning-campus.th-rosenheim.de/course/view.php?id=9266#>

## 2.4 Anschlußbelegung

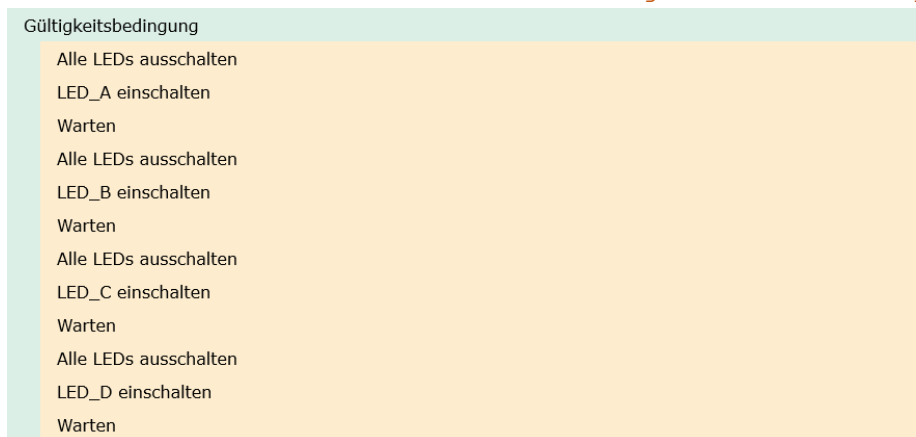
MSP430 LaunchPad	Schrittmotorplatine
P2.0	IN1
P2.1	IN2
P2.2	IN3
P2.3	IN4
GND	-
VCC	+



### 3 Aufgabe 01: Lauflicht für Anfänger

Zunächst wollen wir den Schrittmotor noch nicht mit der Treiberplatine verbinden.  
Wir erzeugen ein Lauflicht an den LEDs der Platine.

1. Erstellen Sie ein neues Projekt in CCS
2. Nennen Sie das die Datei mit dem Hauptprogramm  
**main\_02\_01.c**
3. Konfigurieren Sie die Pins P2.0 bis P2.3 des MSP als Outputs
4. Erzeugen Sie ein Lauflicht. Die LEDs sollen in der Reihenfolge A - B - C - D leuchten.  
Orientieren Sie sich dabei an folgendem Struktogramm:



Anmerkung: Zum Ausschalten der LEDs empfiehlt es sich, eine Variable mit einer Maske zu definieren, in der alle verwendeten Ausgänge definiert sind:

```
const unsigned char c_u8Bestromungsmaske = (LED_A + LED_B + LED_C + LED_D);  
// Alle Pins, die zum Schalten der LEDs verwendet werden
```

## 4 Aufgabe 02: Lauflicht für fortgeschrittene Programmierer

### 4.1 Aufgabenstellung

Wir wollen die Aufgabe [3.](#) etwas eleganter und flexibler lösen.

Hierzu wollen wir die Ansteuerung der LEDs in einem Array speichern:

```
const unsigned char Au8Bestromungsmuster[4] = {LED_A, LED_B, LED_C, LED_D};
```

In unserer Schleife indizieren wird ein Element von **Au8Bestromungsmuster** nach dem anderen und geben dessen Inhalt an Port 2 aus

1. Erzeugen Sie eine neue Datei mit dem Namen **main\_02\_02.c** und übernehmen Sie den Inhalt von **main\_02\_01.c**.
2. Ändern Sie den Quellcode so, daß **Au8Bestromungsmuster** verwendet wird

## 5 Aufgabe 03: Lauflicht ändert Richtung auf Tastendruck

### 5.1 Aufgabenstellung

Erzeugen Sie eine neue Datei **main\_02\_03.c** und erweitern Sie dort Ihr Programm so, daß das Lauflicht seine Richtung ändert, wenn die Taste S2 gedrückt wird.

Verwenden Sie eine Variable **u8Direction**, in der die Drehrichtung gespeichert wird. Mit jedem Druck auf Taste S2 ändert sich der Inhalt der Variablen.

### 5.2 Musterlösung

```

main_01_01.c  main_02_02.c  main_01_02.c  main_01_03.c  main_01_04.c  main_02_01.c  main_02_03.c x
13 #define SET      |=
14 #define CLR      &=~
15
16 #define LED_A      0x01
17 #define LED_B      0x02
18 #define LED_C      0x04
19 #define LED_D      0x08
20
21 #define SWITCH_S2  0x08
22
23 const unsigned char Au8Bestromungsmuster[4] = {LED_A, LED_B, LED_C, LED_D};
24 const unsigned char c_u8Bestromungsmaske   = (LED_A + LED_B + LED_C + LED_D);
25
26 /**
27  * main.c
28  */
29 int main(void)
30 {
31     unsigned int u32CountMax = 60000; // Maximaler Zählerstand der Verzögerungsschleife
32     unsigned int u32_i; // Variable für Verzögerungsschleife
33     unsigned char u8_j = 0; // Variable zur Auswahl des Bestromungsmusters
34     unsigned char u8Direction = 0; // Wenn das LSB von u8Direction = 0 -> die LEDs leuchten aufsteigen, sonst abfallend
35
36     WDCTL = WDTPW | WDTHOLD; // stop watchdog timer
37
38     P1DIR CLR SWITCH_S2; // Pin des Schalters S2 als Input
39     P1REN SET SWITCH_S2; // Interner Pullup/Pulldown enabled
40     P1OUT SET SWITCH_S2; // Interner Pullup enabled
41
42     P2DIR SET (LED_A + LED_B + LED_C + LED_D); // Pins an der Treiberplatine als Ausgänge
43
44     while (1)
45     {
46         P2OUT CLR (c_u8Bestromungsmaske); // Alle LEDs ausschalten
47         P2OUT SET (Au8Bestromungsmuster[u8_j]); // Nächste LED einschalten
48         for (u32_i = 0; u32_i < u32CountMax; u32_i++) // Verzögerung
49         {
50             //NOP();
51         }
52         if (u8Direction & 0x01 == 0x01)
53         {
54             u8_j++; // Zähler erhöhen
55         }
56         else
57         {
58             u8_j--; // Zähler erniedrigen (andere Drehrichtung)
59         }
60         u8_j &= 0x03; // alle Bit außer die niedrigsten 2 löschen
61         if ((P1IN & SWITCH_S2) == 0) // Schalter gedrückt?
62         {
63             u8Direction++; // Richtung umkehren
64             while ((P1IN & SWITCH_S2) == 0) // Warten, bis der Tast wieder losgelassen wurde.
65             {
66                 // NOP();
67             }
68         }
69     }
70     return 0;
71 }

```

## 6 Aufgabe 04: Ansteuerung eines Schrittmotors im Vollschrittbetrieb und im Halbschrittbetrieb

### 6.1 Aufgabenstellung

Jetzt wollen wir einen Schrittmotor ansteuern.

Die Ansteuerung soll zunächst im Vollschrittbetrieb erfolgen.

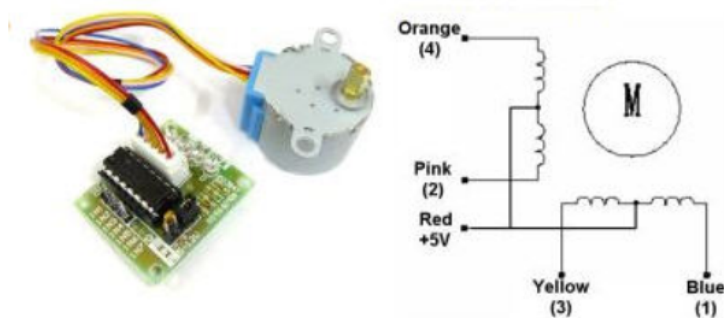
Nach erfolgreicher Inbetriebnahme wollen wir den Schrittmotor auch im Halbschrittbetrieb ansteuern.

Die zur Ansteuerung des nötigen Bestromungsmuster sollen in dem Array

**Au8Bestromungsmuster[]** gespeichert werden.

### 6.2 Informationssammlung

The motor has 4 coils of wire that are powered in a sequence to make the magnetic motor shaft spin. When using the full-step method, 2 of the 4 coils are powered at each step. The default stepper library that comes pre-installed with the Arduino IDE uses this method. The 28BYH-48 datasheet specifies that the preferred method for driving this stepper is using the half-step method, where we first power coil 1 only, then coil 1 and 2 together, then coil 2 only and so on... With 4 coils, this means 8 different signals, like in the table below.



### Half-Step Switching Sequence

Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 Orange	-	-						-
3 Yellow		-	-	-				
2 Pink				-	-	-		
1 Blue						-	-	-

### Wiring the ULN2003 stepper motor driver to Arduino Uno

<http://42bots.com/tutorials/28byj-48-stepper-motor-with-uln2003-driver-and-arduino-uno/>



Für den Vollschrittbetrieb gilt:

Für eine kontinuierliche Drehbewegung muss sich die Ansteuerfolge ständig wiederholen. Das so aus einem Zeigerdiagramm abgeleitete Bestromungsmuster lässt sich in ein Bestromungsdiagramm für die Wicklungen A, A', B und B' übernehmen.

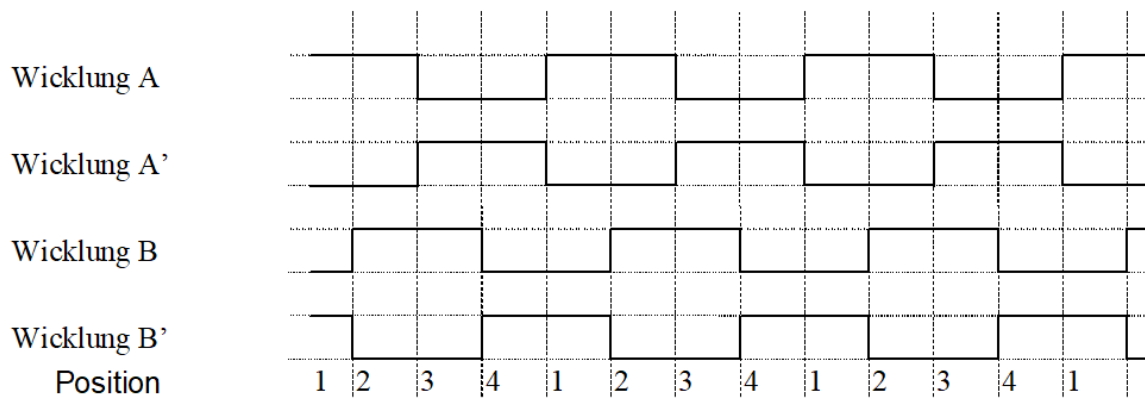


Abbildung 13: Bestromungsdiagramm für Vollschrittbetrieb

### 6.3 Umsetzung

1. Erzeugen Sie eine neue Datei `main_02_04.c` welche auf `main_02_03.c` aufbaut.
2. Verändern Sie den Inhalt von `Au8Bestromungsmuster[]` so, daß die Variable ein Bestromungsmuster für den **Vollschrittbetrieb** enthält.
3. Erstellen Sie anschließend ein Programm `main_02_05.c`, in dem sie den Motor im Halbschrittbetrieb ansteuern.