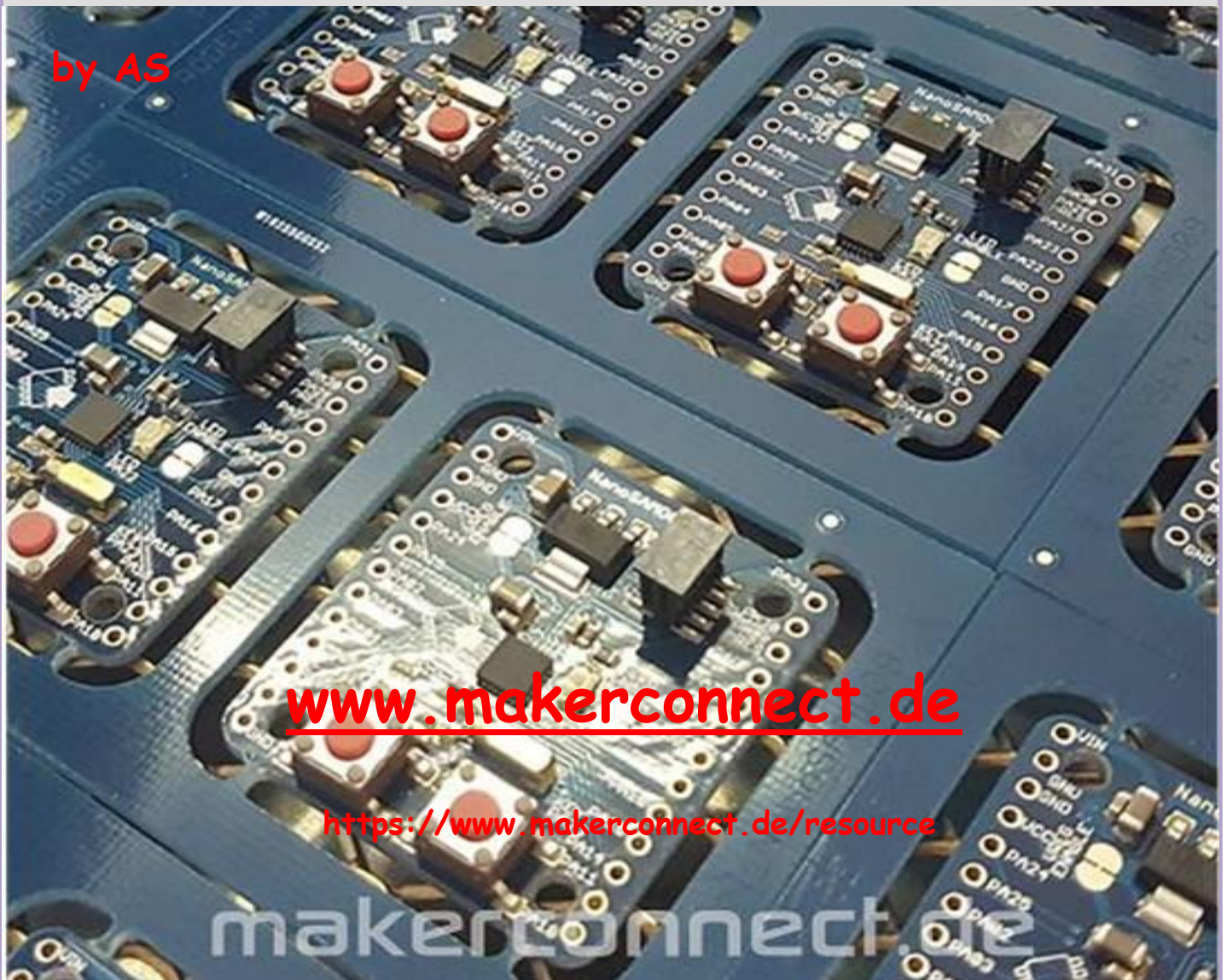


MIKROKONTROLLER & I²C BUS

by AS

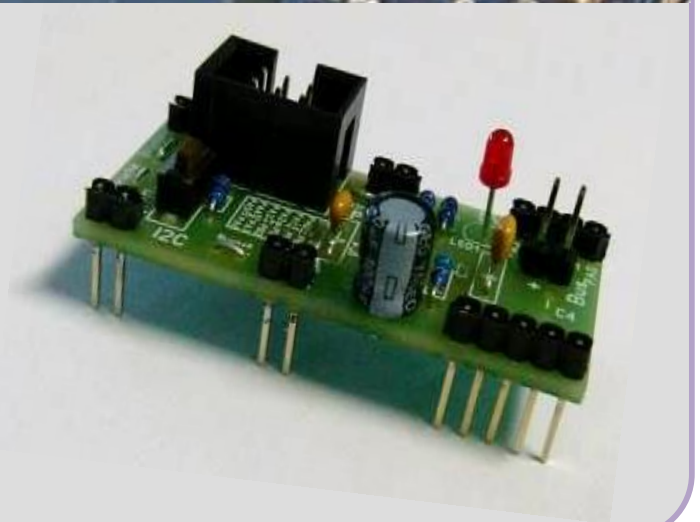


www.makerconnect.de

<https://www.makerconnect.de/resource>

Attiny 841 - Servo
Teil 4 - Timer

I²C Bus und der
Attiny 841



Copyright

Sofern nicht anders angegeben, stehen die Inhalte dieser Dokumentation unter einer „Creative Commons - Namensnennung-NichtKommerziell-Weitergabe unter gleichen Bedingungen 3.0 DE Lizenz“



Sicherheitshinweise

Lesen Sie diese Gebrauchsanleitung, bevor Sie diesen Bausatz in Betrieb nehmen und bewahren Sie diese an einem für alle Benutzer jederzeit zugänglichen Platz auf. Bei Schäden, die durch Nichtbeachtung dieser Bedienungsanleitung verursacht werden, erlischt die Gewährleistung/Garantie. Für Folgeschäden übernehmen wir keine Haftung! Bei allen Geräten, die zu ihrem Betrieb eine elektrische Spannung benötigen, müssen die gültigen VDE-Vorschriften beachtet werden. Besonders relevant sind für diesen Bausatz die VDE-Richtlinien VDE 0100, VDE 0550/0551, VDE 0700, VDE 0711 und VDE 0860. Bitte beachten Sie auch nachfolgende Sicherheitshinweise:

- Nehmen Sie diesen Bausatz nur dann in Betrieb, wenn er zuvor berührungssicher in ein Gehäuse eingebaut wurde. Erst danach darf dieser an eine Spannungsversorgung angeschlossen werden.
- Lassen Sie Geräte, die mit einer Versorgungsspannung größer als 24 V- betrieben werden, nur durch eine fachkundige Person anschließen.
- In Schulen, Ausbildungseinrichtungen, Hobby- und Selbsthilfwerkstätten ist das Betreiben dieser Baugruppe durch geschultes Personal verantwortlich zu überwachen.
- In einer Umgebung in der brennbare Gase, Dämpfe oder Stäube vorhanden sind oder vorhanden sein können, darf diese Baugruppe nicht betrieben werden.
- Im Falle einer Reparatur dieser Baugruppe, dürfen nur Original-Ersatzteile verwendet werden! Die Verwendung abweichender Ersatzteile kann zu ernsthaften Sach- und Personenschäden führen. Eine Reparatur des Gerätes darf nur von fachkundigen Personen durchgeführt werden.
- Spannungsführende Teile an dieser Baugruppe dürfen nur dann berührt werden (gilt auch für Werkzeuge, Messinstrumente o.ä.), wenn sichergestellt ist, dass die Baugruppe von der Versorgungsspannung getrennt wurde und elektrische Ladungen, die in den in der Baugruppe befindlichen Bauteilen gespeichert sind, vorher entladen wurden.
- Sind Messungen bei geöffnetem Gehäuse unumgänglich, muss ein Trenntrafo zur Spannungsversorgung verwendet werden
- Spannungsführende Kabel oder Leitungen, mit denen die Baugruppe verbunden ist, müssen immer auf Isolationsfehler oder Bruchstellen kontrolliert werden. Bei einem Fehler muss das Gerät unverzüglich ausser Betrieb genommen werden, bis die defekte Leitung ausgewechselt worden ist.
- Es ist auf die genaue Einhaltung der genannten Kenndaten der Baugruppe und der in der Baugruppe verwendeten Bauteile zu achten. Gehen diese aus der beiliegenden Beschreibung nicht hervor, so ist eine fachkundige Person hinzuzuziehen

Bestimmungsgemäße Verwendung

- Auf keinen Fall darf 230 V~ Netzspannung angeschlossen werden. Es besteht dann Lebensgefahr!
- Dieser Bausatz ist nur zum Einsatz unter Lern- und Laborbedingungen konzipiert worden. Er ist nicht geeignet, reale Steuerungsaufgaben jeglicher Art zu übernehmen. Ein anderer Einsatz als angegeben ist nicht zulässig!
- Der Bausatz ist nur für den Gebrauch in trockenen und sauberen Räumen bestimmt.
- Wird dieser Bausatz nicht bestimmungsgemäß eingesetzt kann er beschädigt werden, was mit Gefahren, wie z.B. Kurzschluss, Brand, elektrischer Schlag etc. verbunden ist. Der Bausatz darf nicht geändert bzw. umgebaut werden!
- Für alle Personen- und Sachschäden, die aus nicht bestimmungsgemäßer Verwendung entstehen, ist nicht der Hersteller, sondern der Betreiber verantwortlich. Bitte beachten Sie, dass Bedien- und /oder Anschlussfehler außerhalb unseres Einflussbereiches liegen. Verständlicherweise können wir für Schäden, die daraus entstehen, keinerlei Haftung übernehmen.
- Der Autor dieses Tutorials übernimmt keine Haftung für Schäden. Die Nutzung der Hard- und Software erfolgt auf eigenes Risiko.

ATTiny 841 - Servo Teil 4 - Timer

Es geht auch einfacher. Sehen wir uns noch mal das vereinfachte Diagramm an.

- Es wird alle 20 ms ein Impuls benötigt
- Dieser Impuls sollte eine Breite von 1,0 bis 2,0 ms haben

Damit haben wir die Grundlagen zur Ansteuerung eines Servos bereits fertig.

Der Attiny 841 hat insgesamt 3 Timer

- Timer 0 - 8 Bit
- Timer 1 - 16 Bit
- Timer 2 - 16 Bit

In unserem Beispiel werde ich den Timer 1 verwenden. Im Datenblatt zum Attiny 841 wird im Abschnitt 12.1, in den technischen Merkmalen, angegeben, dass er zwei Register hat, die unabhängig geschaltet werden können.

Zunächst brauchen wir aber einen Impuls der alle 20 ms schaltet.

In diesem Tut möchte ich euch nicht mit langweiligen Berechnungen quälen. Es gibt im Netz verschiedene Programme zur Berechnung der Daten.

Einstellung der Ausgangsdaten:

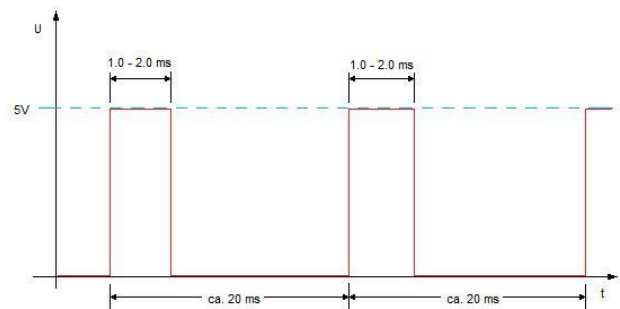
$f_{osz} = 8 \text{ MHz}$ (8 000 000 Hz)

Zeit = 20 ms (20 000 μs)

Timer 1 = 16 Bit

Compare Mode

Damit ergeben sich diese Ergebnisse:



Frequenz [Hz]

Interrupt
☐ Overflow (Reload)
☒ Compare

Timer
☒ 16 Bit
☐ 8 Bit

Interrupt-Time [μs]

Compare-Interrupt
Bei einem Compare-Ereignis wird der Compare-Interrupt ausgelöst, im CTC-Mode wird der Timer automatisch auf 0x0000 zurückgesetzt.

Prescaler	Compare	Time [μs]	Error [μs]
1	-	-	-
2	-	-	-
4	39999	20000	0,000
8	19999	20000	0,000
16	9999	20000	0,000
32	4999	20000	0,000
64	2499	20000	0,000
128	1249	20000	0,000
256	624	20000	0,000

Je nach verwendeten Attiny sind unterschiedliche **Prescaler** (Vorteiler) möglich. Nach einem Blick in das Datenblatt des Attiny 841 werde ich den **Prescaler = 8** verwenden.

Aus der **Tabelle 12-6** wähle ich die Einstellung für einen **Prescaler = 8** aus:

`TCCR1B|=(1<<CS11);` // Prescaler 8 bei 8MHz

Aus der **Tabelle 12-5** (Waveform Generation Mode) wähle ich den Mode 4 (CTC) aus:

`TCCR1B|=(1<<WGM12);` // CTC Modus Nr. 4

Mit TIMSK1 werden beide Register eingeschaltet/erlaubt.

`TIMSK1|=(1<<OCIE1A)|(1<<OCIE1B);` // Interrupt erlauben

Bei einer angegebenen Frequenz von 8 MHz und einem Prescaler = 8 muss ein Compare von 19999 angegeben werden. Damit ergibt sich eine Zeit von genau 20 ms.

Mit OCR1A nehme ich die Einstellung für Compare vor:

```
OCR1A=19999; // Timer 1 Register auf 19999 = 20ms bei 8 MHz
```

Die Einstellung für OCR1B gebe ich im Programm an bzw. wird berechnet.

```
OCR1B=stellung; // Timer 1 Register B Mittelstellung
```

Damit ergibt sich für den Timer1 dieses Programm:

```
void timer1_init()
{
    TCCR1B|= (1<<CS11); // Timer 1 16 Bit konfigurieren
    TCCR1B|= (1<<WGM12); // Prescaler 8 bei 8MHz
    TIMSK1|= (1<<OCIE1A)|(1<<OCIE1B); // CTC Modus Nr 4
    OCR1A=19999; // Interrupt erlauben
    OCR1B=stellung; // Timer 1 Register auf 19999 = 20ms bei 8 MHz
    // Timer 1 Register B Mittelstellung
}
```

Die Ausgabe der Impulse erfolgt am PIN A5

```
ISR(TIMER1_COMPA_vect) // Impuls starten
{
    PORTA |= (1<<PINA5);
}
ISR(TIMER1_COMPB_vect) // Impuls stoppen
{
    PORTA &= ~(1<<PINA5);
}
```

Die Auswahl der Stellungen des Servos erfolgt durch die Taster 1-3. Musste dabei feststellen das Taster relativ stark prellen. Deshalb habe ich diese einfache Entprellung eingefügt.

```
if (!(taste1)) // Taste 1
{
    _delay_ms(20); // entprellen
    if (!(taste1)) // Taste 1
    {
        OCR1B = stellungmin; // Stellung minimal
    }
}
```

Durch die Definition der Taster und der Werte kann ich alle notwendigen Einstellungen am Anfang des Programmes vornehmen.

```
#define taste1 (PINA & (1<<PINA1)) // Taste 1
#define taste2 (PINA & (1<<PINA2)) // Taste 2
#define taste3 (PINA & (1<<PINA3)) // Taste 3
uint16_t stellung = 1000; // Stellung beim einschalten
uint16_t stellungmax = 2200; // Stellung Maximal Taste 1
uint16_t stellungmitt = 1500; // Stellung Mitte Taste 2
uint16_t stellungmin = 580; // Stellung Minimal Taste 3
```

Im zweiten Programm kann ich mit den Tasten 1 und 3 den Servo nach links oder rechts drehen lassen bis zum Endanschlag. Mit der Taste 2 kann ich eine Mittelstellung auswählen.

```
if (!(taste1))                // drehen mit Taste 1
{
    _delay_ms(20);            // Entprellung
    if (!(taste1))
    {
        if(OCR1B < stellungmax) // Drehung berechnen
        {
            OCR1B = OCR1B + 20;
            if(OCR1B >= stellungmax) // Begrenzung
            {
                OCR1B = stellungmax;
            }
        }
        _delay_ms(50);
    }
}
```

Die angegebenen Werte für **stellung**, **stellungmax**, **stellungmitt** und **stellungmin** sind bei jedem Servo unterschiedlich. Habe die Werte an drei verschiedenen Servos unterschiedlicher Hersteller getestet. Entweder drehen sie links oder rechts oder die Endanschläge sind unterschiedlich. Kommt es am Anfang oder Ende des Bereiches zu einem „zittern oder knurren“ sind die Werte unbedingt anzupassen.

Als Hardware habe ich das Mini Board 2 und ein 3-fach Taster Board verwendet. Beim Betrieb ist ein gutes Netzteil mit ausreichender Belastung zu wählen. Sonst kann es beim Schalten des Servos zu einem kurzen Spannungseinbruch und damit einem Reset der Schaltung führen.

In diesem Tut möchte ich zwei verschiedene Programme vorstellen. Im ersten Programm sind feste Zeiten eingestellt. Dadurch kann man schnell überprüfen ob der Servo korrekt funktioniert. Im zweiten Programm kann ich den Bereich durch betätigen der Taster langsam durchfahren. Zum Anfang habe ich die genutzten Pins definiert und die Fuse Einstellungen

// Programm 1 - Platine P148 Attiny 841 ohne Quarz, Servo mit fester Zeit, Ausgänge:

```
// PA1 --> T1
// PA2 --> T2
// PA3 --> T3
// PA5 --> Servo PWM
// Fuse Einstellung ohne Quarz
// Ex - 0xFF
// Hi - 0xDF
// Lo - 0xC2 // C = CLKO aus, ohne Quartz

#define F_CPU 8000000UL        // Angabe der Frequenz, wichtig für die Zeit
#include "avr/io.h"
#include "util/delay.h"
#include "avr/interrupt.h"

#define taste1 (PINA & (1<<PINA1)) // Taste 1
```

```

#define taste2 (PINA & (1<<PINA2)) // Taste 2
#define taste3 (PINA & (1<<PINA3)) // Taste 3

uint16_t stellung = 1000; // Stellung beim einschalten
uint16_t stellungmax = 2200; // Stellung Maximal Taste 1
uint16_t stellungmitt = 1500; // Stellung Mitte Taste 2
uint16_t stellungmin = 580; // Stellung Minimal Taste 3

void timer1_init()
{
    // Timer 1 16 Bit konfigurieren
    TCCR1B |= (1<<CS11); // Prescaler 8 bei 8MHz
    TCCR1B |= (1<<WGM12); // CTC Modus Nr 4
    TIMSK1 |= (1<<OCIE1A)|(1<<OCIE1B); // Interrupt erlauben
    OCR1A = 19999; // Timer 1 Register auf 19999 = 20ms bei 8 MHz
    OCR1B = stellung; // Timer 1 Register B Mittelstellung
}
ISR(TIMER1_COMPA_vect) // Impuls starten
{
    PORTA |= (1<<PINA5);
}
ISR(TIMER1_COMPB_vect) // Impuls stoppen
{
    PORTA &= ~(1<<PINA5);
}
int main(void)
{
    DDRA |= 0b00100000 ; // PA5 auf Ausgang
    sei(); // Interrupts deaktiviert
    timer1_init();
    while(1)
    {
        if (!(taste1)) // Taste 1
        {
            _delay_ms(20); // entprellen
            if (!(taste1)) // Taste 1
            {
                OCR1B = stellungmin; // Stellung minimal
            }
        }
        if (!(taste2)) // Taste 2
        {
            _delay_ms(20); // entprellen
            if (!(taste2)) // Taste 2
            {
                OCR1B = stellungmitt; // Stellung mitte
            }
        }
    }
}

```

```

    if (!(taste3))                // Taste 3
    {
        _delay_ms(20);            // entprellen
        if (!(taste3))            // Taste 3
        {
            OCR1B = stellungmax;   // Stellung maximal
        }
    }
}

```

Kommen wir zum zweiten Programm:

```

// Programm 2 - Platine P148 Attiny 841 ohne Quarz, Servo mit veränderliche Zeit
#define F_CPU 8000000UL           // Angabe der Frequenz, wichtig für die Zeit
#include "avr/io.h"
#include "util/delay.h"
#include "avr/interrupt.h"

#define taste1 (PINA & (1<<PINA1)) // Taste 1 links drehen
#define taste2 (PINA & (1<<PINA2)) // Taste 2 Mittelstellung
#define taste3 (PINA & (1<<PINA3)) // Taste 3 rechts drehen

uint16_t stellung = 1000;         // Stellung Mitte mit Taster 2
uint16_t stellungmax = 2180;      // Stellung Maximal Taste 1
uint16_t stellungmitt = 1500;     // Stellung beim einschalten
uint16_t stellungmin = 570;       // Stellung Minimal Taste 3

void timer1_init()
{
    // Timer 1 16 Bit konfigurieren
    TCCR1B |= (1<<CS11);           // Prescaler 8 bei 8MHz
    TCCR1B |= (1<<WGM12);          // CTC Modus Nr 4
    TIMSK1 |= (1<<OCIE1A)|(1<<OCIE1B); // Interrupt erlauben
    OCR1A = 19999;                 // Timer 1 Register A auf 19999 = 20ms
    OCR1B = stellungmitt;          // Timer 1 Register B Mittelstellung
}

ISR(TIMER1_COMPA_vect)             // Impuls starten
{
    PORTA |= (1<<PINA5);
}

ISR(TIMER1_COMPB_vect)             // Impuls stoppen
{
    PORTA &= ~(1<<PINA5);
}

int main(void)
{
    DDRA |= 0b00100000;           // PA5 auf Ausgang
    sei();                         // Interrupts deaktiviert
    timer1_init();
    while(1)

```

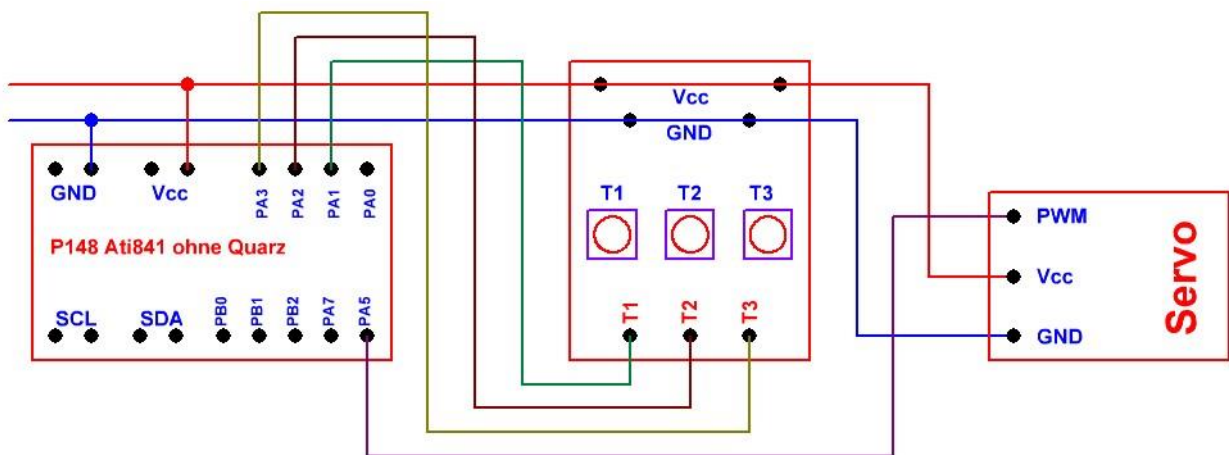
```

{
  if (!(taste1))                                // drehen mit Taste 1
  {
    _delay_ms(20);                               // Entprellung
    if (!(taste1))
    {
      if(OCR1B < stellungmax)                     // Drehung berechnen
      {
        OCR1B = OCR1B + 20;
        if(OCR1B >= stellungmax)                 // Begrenzung
        {
          OCR1B = stellungmax;
        }
      }
      _delay_ms(50);
    }
  }
  if (!(taste2))                                // Auswahl Mittelstellung mit Taste 2
  {
    _delay_ms(20);                               // Entprellung
    if (!(taste2))
    {
      OCR1B = stellung;                          // Auswahl Mittelstellung
    }
  }
  if (!(taste3))                                // drehen mit Taste 3
  {
    _delay_ms(20);                               // Entprellung
    if (!(taste3))
    {
      if(OCR1B > stellungmin)                     // Drehung berechnen
      {
        OCR1B = OCR1B - 20;
        if(OCR1B <= stellungmin)                 // Begrenzung
        {
          OCR1B = stellungmin;
        }
      }
      _delay_ms(50);
    }
  }
}
}

```

Bitte unbedingt **stellung**, **stellungmax**, **stellungmitt** und **stellungmin** anpassen. Jedes Servo, je nach Hersteller oder Bauart, kann anders reagieren. Bitte die Stromaufnahme kontrollieren oder ein separates Netzteil verwenden.

Verwendete Schaltung mit Mini Board 2, Taster Board und Servo:



Einige Teile des Textes wurden zur besseren Übersicht farblich gestaltet.
Die Nutzung erfolgt auf eigenes Risiko.

Ich wünsche viel Spaß beim Bauen und programmieren
Achim

Quellenangabe:

https://www.mikrocontroller.net/articles/AVR-Tutorial:_Servo

https://www.mikrocontroller.net/articles/Modellbauservo_Ansteuerung

verschiedene Artikel aus

<https://www.mikrocontroller.net> und

<https://rn-wissen.de/wiki/index.php/Hauptseite>

<https://wolles-elektronikkiste.de/timer-und-pwm-teil-2-16-bit-timer1>

Programmen von Peter Danegger, Stefan Frinks und oberallgeier