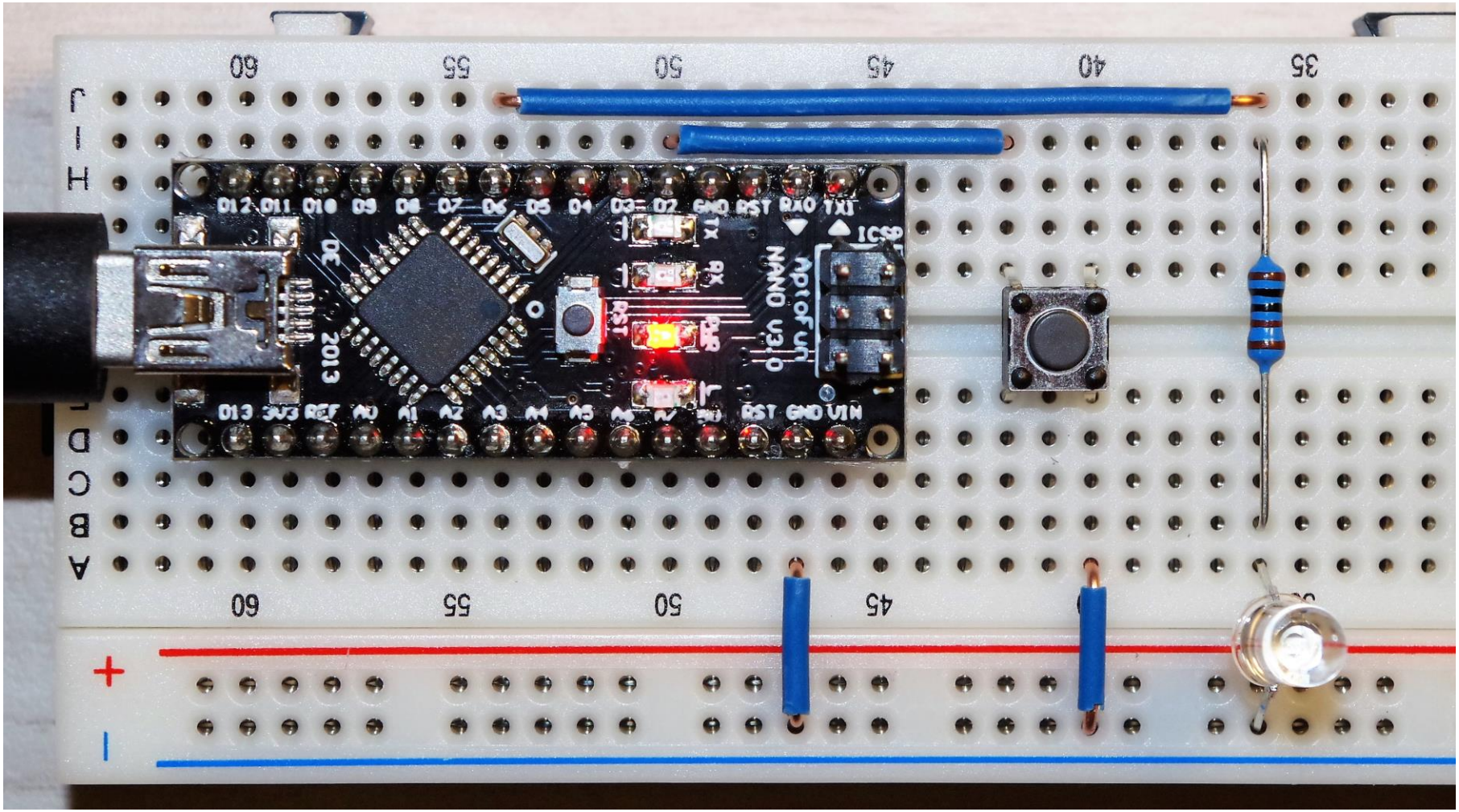
Mikrocontrollerplatine vorbereiten

Aufgabe B1

Verbinden Sie einen Tastschalter mit dem Anschluss PD2 und eine Leuchtdiode mit dem Anschluss PD6 (Vorwiderstand nicht vergessen!).



Mikrocontrollerplatine testen

Aufgabe B1

Laden Sie das folgende Testprogramm auf den Mikrocontroller. Wenn der Taster gedrückt wird, blinken die interne und die externe Leuchtdiode.

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

void led_on(void)
{
  PORTB = 0b00100000;
  PORTD = 0b01000100; // Pullup an PD2!
}

void led_off(void)
{
  PORTB = 0b00000000;
  PORTD = 0b00000100; // Pullup an PD2!
}

int taster_gedrueckt(void)
{
  if((PIND & 0b00000100) == 0) return 1;
  return 0;
}
```

```
int main(void)
{
  DDRB  = 0b00100000; // LED/PB5
  DDRD  = 0b01000000; // LED/PD6
  PORTD = 0b00000100; // Pullup/PD2

  while (1)
  {
    if(taster_gedrueckt() == 1)
    {
      led_on();
      _delay_ms(100);

      led_off();
      _delay_ms(100);
    }
  }
}
```

Anmerkungen zur Implementierung des Testprogramms

Aufgabe B2

- Ein großer Nachteil des „Blinkprogramms“ auf der vorherigen Folie ist, dass die Verzögerungen zum Ein-/Ausschalten der LEDs im Hauptprogramm mittels _delay_ms(100); erzeugt werden.
- Das Hauptprogramm ist während dieser Verzögerungen blockiert und kann in dieser Zeit keine anderen Aktionen durchführen.
- Wesentlich eleganter ist es, wenn der Ablauf des Programms durch einen „Timer“ gesteuert wird, der automatisch dafür sorgt, dass in regelmäßigen Zeitintervallen eine bestimmte Funktion im C-Programm aufgerufen wird.
- Das Hauptprogramm hat dann nichts mehr mit dem Blinken der LEDs zu tun und kann sich währenddessen um ganz andere Dinge kümmern.
- Laden Sie das Programm auf der folgenden Folie auf den Mikrocontroller und überzeugen Sie sich davon, dass die LEDs blinken.
- Laden Sie das Datenblatt des hier verwendeten Mikrocontrollers ATmega328P von der Webseite des Herstellers (www.atmel.com) herunter und versuchen Sie die Initialisierung des Timers in der Funktion init_timer() zu verstehen (Kapitel 19: TC0 – 8-bit Timer/Counter0 with PWM).
- Berechnen Sie die Frequenz, mit der die LEDs blinken.

Erstes Demoprogramm zum Timer/Counter0

Aufgabe B2

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

void led_on(void)
{
  PORTB = 0b00100000;
  PORTD = 0b01000100;   // Pullup/PD2!
}

void led_off(void)
{
  PORTB = 0b00000000;
  PORTD = 0b00000100;   // Pullup/PD2!
}

void init_timer(void)
{
  TCCR0A = 0b00000010; // WGM01 setzen (siehe Tab. 19-9, S. 140)
  TCCR0B = 0b00000101; // CS00+CS02 setzen (Tab. 19-10, S. 142)
  TIMSK0 = 0b00000010; // OCIE0A setzen (siehe S. 143)
  OCR0A  = 255;        // Vergleichswert für Timer
  sei();               // Interrupt-System einschalten
}
```

```
int led = 0; // Aktueller LED-Zustand
ISR(TIMER0_COMPA_vect)
{
  switch(led)
  {
    case 0: led_on();  led = 1; break;
    case 1: led_off(); led = 0; break;
  }
}

int main(void)
{
  DDRB  = 0b00100000;  // LED/PB5
  DDRD  = 0b01000000;  // LED/PD6
  PORTD = 0b00000100;  // Pullup/PD2
  init_timer();
  while(1) { }  // Endlosschleife
}
```

Leuchtdioden blinken mit 0,5 Hz

Aufgabe B3

- Ändern Sie das Programm von der vorherigen Folie, sodass die Leuchtdioden mit einer Frequenz von 0,5 Hz blinken, also eine Sekunde ein, eine Sekunde aus usw…
- Frequenz des Timer-Interrupts = 1 kHz; verwenden Sie folgenden Programmcode:

```
unsigned long millis = 0;  // Millisekunden seit Programmstart
int led = 0;               // Aktueller Zustand der LEDs

ISR(TIMER0_COMPA_vect)
{
    millis++;  // Millisekunden zählen
    if(millis % 1000 == 0)  // LEDs 1x pro Sekunde umschalten
    {
        switch(led)
        {
        case 0: led_on();  led = 1; break;
        case 1: led_off(); led = 0; break;
        }
    }
}
```

- Zusatzaufgabe: Solange der Taster gedrückt ist, sollen die LEDs mit der 10-fachen Frequenz blinken. Erweitern Sie dazu das bestehende Programm.

Umschalten zwischen verschiedenen Frequenzen (a)

Aufgabe B4

Erweitern Sie das bestehende Programm, sodass die Blinkfrequenz durch mehrfachen Tastendruck zwischen 1, 5, 10 und 50 Hz umgeschaltet werden kann:

```
volatile char state = 1;  // Nummer der aktuellen Frequenz (1...4)

int main(void)
{
    // TODO: Ports und Timer initialisieren...
    while(1) // Endlosschleife
    {
        // Wenn der Anwender die Frequenz verändern möchte,
        // wird "state" auf einen neuen Wert gesetzt.
        if(frequenz_weiterschalten() == 1)
        {
            switch(state)
            {
                case 1: state = 2; break;
                case 2: state = 3; break;
                case 3: state = 4; break;
                case 4: state = 1; break;
            }
        }
    }
}
```

Umschalten zwischen verschiedenen Frequenzen (b)

Aufgabe B4

Die Auswertung des Tasters – zur Umschaltung der Frequenz – kann folgendermaßen programmiert werden:

```
int frequenz_weiterschalten(void)
{
    // Taster nicht gedrückt --> Rückgabe == 0.
    if((PIND & 0b00000100) != 0) return 0;

    // Kurz warten, Taster kann prellen...
    _delay_ms(100);

    // Abwarten, bis Taster wieder losgelassen wird...
    while((PIND & 0b00000100) == 0) { }

    // Nochmals kurz warten, Taster kann wieder prellen...
    _delay_ms(100);

    // Frequenz soll verändert werden --> Rückgabe == 1.
    return 1;
}
```