Vorbereitung

So wird der **Analog-Digital-Wandler** des Mikrocontrollers ATmega328P initialisiert:

```
ADMUX = _BV(REFS0);
ADCSRA = _BV(ADEN) | _BV(ADPS2) | _BV(ADPS1) | _BV(ADPS0);
```

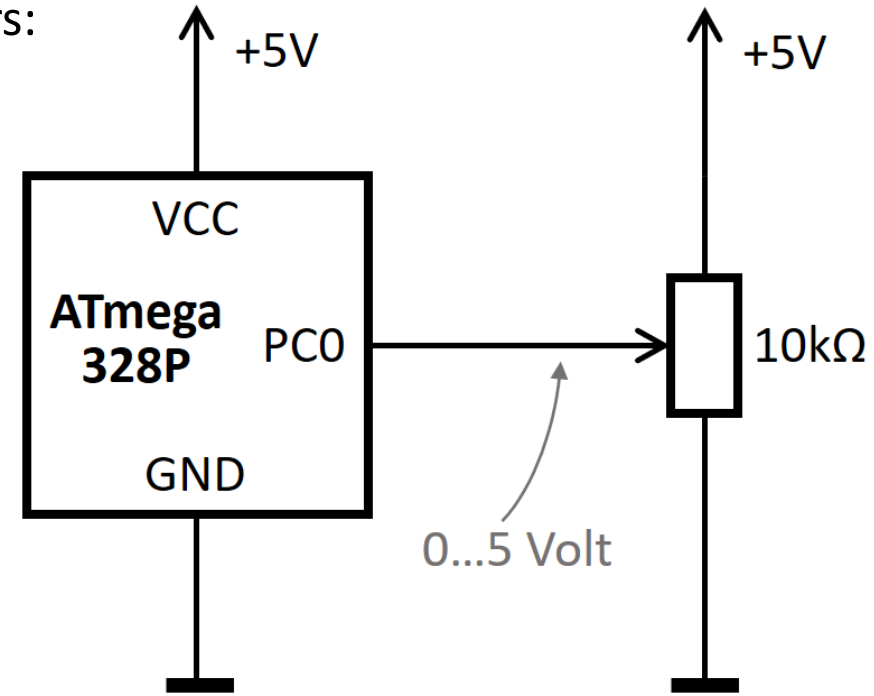
Der Analog-Digital-Wandler ist im Datenblatt in Kapitel 28 („ADC - Analog to Digital Converter“) beschrieben. Beachten Sie insbesondere die Abschnitte 28.4 („Prescaling and Conversion Timing“) sowie 28.9 („Register Description“):

- **An welchem Pin des Controllers wird das Analogsignal angeschlossen?**
- **Wie wird eine Analog-Digital-Wandlung gestartet?**
- **Woran erkennt man, dass die Wandlung beendet wurde?**
- **In welchem Register steht das Ergebnis?**
- Wie lauten die minimalen und maximalen Spannungswerte, die vom Analog-Digital-Wandler noch erfasst werden können?
- Welche Werte werden im Ergebnisregister für die minimal und für die maximal mögliche Eingangsspannung zurückgegeben?
- Wie groß (in Volt) ist der kleinste Spannungsschritt, der erfasst werden kann?
- Wie lange dauert eine einzelne Analog-Digital-Wandlung?

Zweipunktregler mit Hysterese (a)

Aufgabe D1

Verbinden Sie zunächst einen regelbaren Widerstand mit dem Anschluss PC0 des Mikrocontrollers:



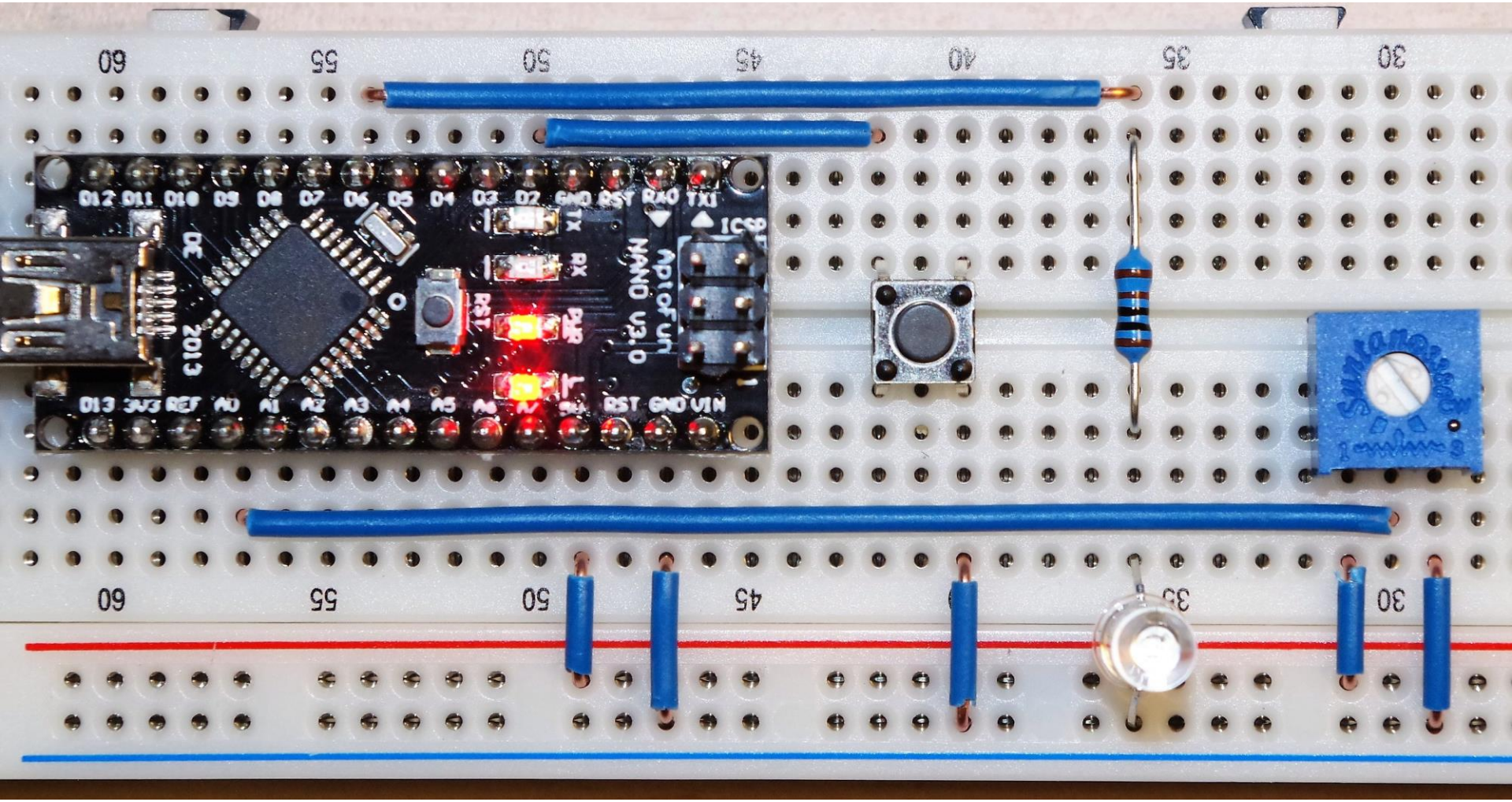
Erstellen Sie anschließend ein Programm mit den folgenden Funktionen:

- Die auf der Mikrocontrollerplatine eingebaute LED (am Anschluss PB5) soll eingeschaltet werden, wenn die Spannung am Eingang PC0 einen Wert von 2,0 Volt überschreitet.
- Die LED soll wieder ausgeschaltet werden, wenn die Spannung an PC0 einen Wert von 1,5 Volt unterschreitet.

Zweipunktregler mit Hysterese (b)

Aufgabe D1

Anschluss des regelbaren Widerstands an ADC0/PC0:



Die Leuchtdiode an PD6 und der Taster an PD2 können angeschlossen bleiben…

Berechnung von Mittelwerten

Aufgabe D2

Die Spannungen am Eingang des Analog-Digital-Wandlers sind in der Regel von einem gewissen Rauschen überlagert. Dadurch kann die Funktion des Reglers negativ beeinflusst werden.

Um den Regler robuster gegenüber solchen Störungen zu machen, erweitern Sie das soeben erstellte Programm wie folgt:

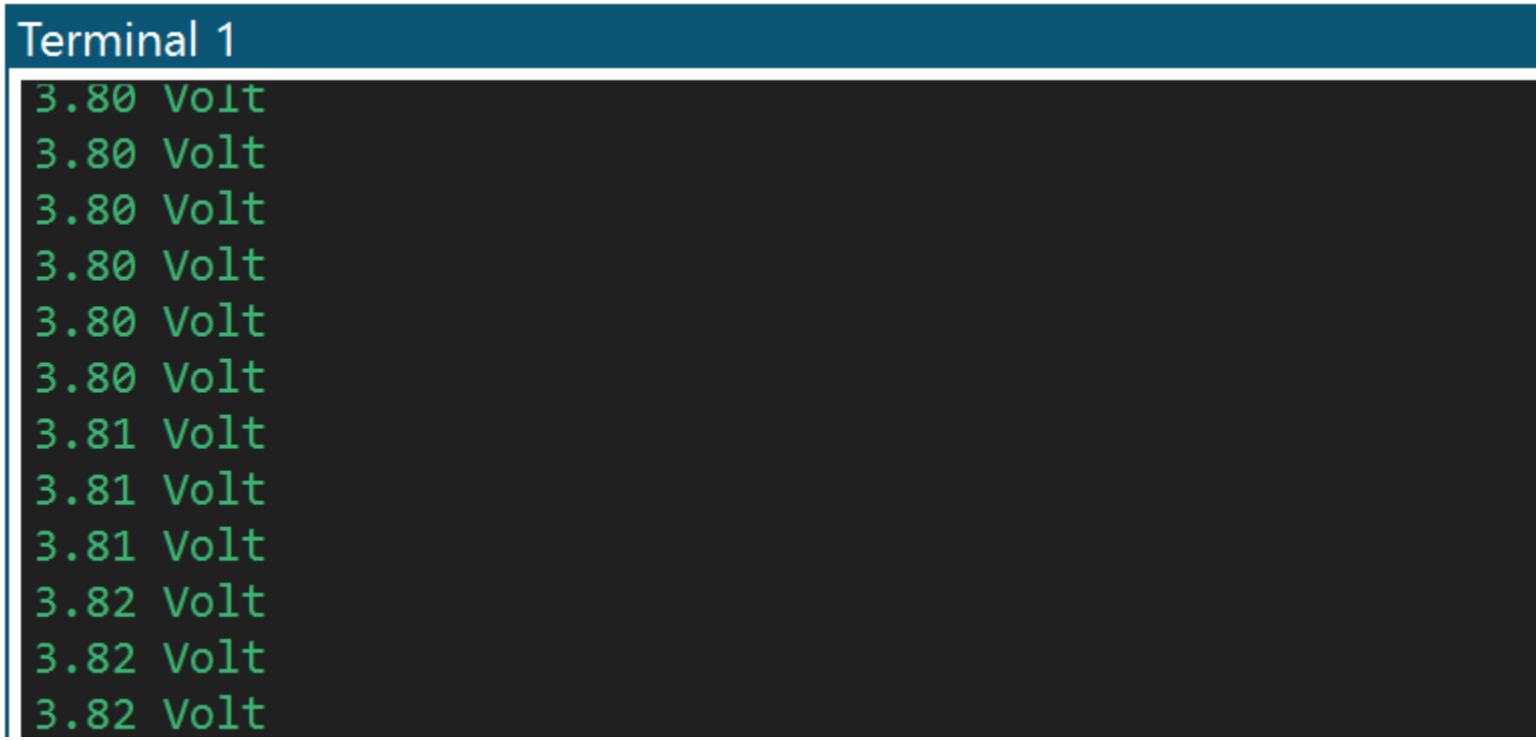
- Es werden stets 8 Analog-Digital-Wandlungen nacheinander durchgeführt und von allen 8 eingelesenen Werten der Mittelwert berechnet.
- Wenn der berechnete Mittelwert mehr als 2,0 Volt bzw. weniger als 1,5 Volt beträgt, schaltet der Ausgang des Reglers wie auf Folie 2 beschrieben um.
- Führen Sie alle Berechnungen ohne Verwendung von Fließkommazahlen (Datentypen float oder double) durch, da Berechnungen mit Fließkommazahlen auf kleinen 8-Bit-Mikrocontrollern einen hohen Aufwand verursachen.

Voltmeter mit serieller Schnittstelle (a)

Aufgabe D3

Erstellen Sie ein Programm, welches die an PC0 eingelesenen Spannungen regelmäßig über die serielle Schnittstelle an den angeschlossenen PC sendet.

- Es sollen exakt 10 Messungen pro Sekunde durchgeführt werden.
- Damit die Messungen zum richtigen Zeitpunkt stattfinden, starten Sie die einzelnen Analog-Digital-Wandlungen mithilfe eines Timer-Interrupts.



Voltmeter mit serieller Schnittstelle (b)

Aufgabe D3

Tipps zur Programmierung:

- Erstellen Sie ein neues Projekt und übernehmen Sie die folgenden Funktionen aus bereits bestehenden Programmen:
  - Initialisierung des Timers, Timer-Interrupt, Initialisierung des Analog-Digital-Wandlers, Mittelwertbildung von Analog-Digital-Wandlungen, Initialisierung der seriellen Schnittstelle, Senden von einzelnen Zeichen und ganzen Zeichenketten inkl. Zeilenumbruch.
- Stellen Sie den Timer so ein, dass 1000 Interrupts pro Sekunde ausgelöst werden und zählen Sie die abgelaufenen Millisekunden im Timer-Interrupt. Alle 100 Millisekunden wird im Timer-Interrupt eine globale volatile-Variable mit dem Namen „want_adc“ auf 1 gesetzt.
- Nach der Initialisierung von Analog-Digital-Wander, Schnittstelle und Timer wartet das Hauptprogramm darauf, dass „want_adc“ auf 1 wechselt. Anschließend wird die Analog-Digital-Wandlung durchgeführt, der eingelesene Wert in eine Zeichenkette konvertiert und über die serielle Schnittstelle zum PC übertragen.
- Nun setzt das Hauptprogramm die Variable „want_adc“ wieder auf 0 und wartet solange, bis eine neue Analog-Digital-Wandlung angefordert wird…

Voltmeter mit serieller Schnittstelle (c)

Aufgabe D3

```
int main(void)
{
    char msg[] = "0.00 Volt";

    USART_Init(MYUBRR);
    init_port_and_adc();
    init_timer();

    while(1)
    {
        if(want_adc)
        {
            want_adc = 0;
            uint32_t analog = read_adc0_mean();
            analog *= 500, analog /= 1023;

            msg[3] = '0' + analog % 10; analog /= 10;
            msg[2] = '0' + analog % 10; analog /= 10;
            msg[0] = '0' + analog % 10;
            USART_TransmitLine(msg);
        }
    }
}
```

Was passiert hier eigentlich genau…?