Lösung 06: Analoge Eingabe

Aufgabe 1: Vorbereitung, Schaltung

a) Der ATmega2560 verfügt über 16 Pins für die analoge Eingabe: ADC0 („Analog Pin 0“) bis ADC15 („Analog Pin 15“). Nur über diese Pins kann man analoge Spannungen einlesen. Diese Pins gehören alle zu den Ports F und K des Mikrocontroller (Datenblatt, Seite 2 oder Seite 268 oder Seite 277).

b) Man schlägt im Pin Mapping nach, dass „Analog Pin 2“ des Arduino Boards mit ADC2/PF2 verbunden ist. PF2 ist der 3. Pin („Pin Nummer 2“) des Ports F.

c) Bei mittiger Einstellung liegt am Pin ADC2 die Spannung 2,5V an. Der interne A/D Umsetzer berechnet näherungsweise den Wert: $\frac{2{,}5V}{5.0V} \cdot 2^{10} = 512$. Eine Spannung von 2,5V wird im Mikrocontrollerprogramm also durch den Wert 512 repräsentiert.

Aufgabe 2: Analoger Eingang mit Arduino Library

Nutzt man den *Analog Pin x*, so muss man auch das Kommando `analogRead(x)` verwenden.

```
void setup()
{
	Serial.begin(9600);
}

void loop()
{
	int read = analogRead(2);

	double result = 5.0 * read / 1024.0;
	Serial.print(result);
	Serial.println(" Volt");

	delay(1000);

}
```

Aufgabe 3: Analoger Eingang mit AVR-Libc

a) Laut Handbuch benötigt der A/D Umsetzer eine Eingangsfrequenz von 50 bis 200 kHz. Die 16 Mhz müssen durch den Prescaler in diesen Bereich gebracht werden. Man überzeugt sich durch Rechnung, dass ein /128-Prescaler benötigt wird, um in diesen Bereich zu kommen.
16 MHz / 128 = 125 kHZ
Es gäbe einen /2, /4, /8, /16, /32, /64 und /128 Prescaler.

b) Hinweis: Wichtig ist, dass man ADCL und ADCH in der richtigen Reihenfolge ausliest. Zuerst muss ADCL gelesen werden, dann ADCH, siehe Seite 270 (3. Abschnitt).

```
void setup()
{
	// activate serial console
	Serial.begin(9600);

	// enable ADC functionality
	ADCSRA |= (1 << ADEN);

	// use /128 prescaler (ADC requires 50 kHz to 200 kHz, see manual p271, but
system clock is 16 MHz)
	ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

	// select ADC2 as input pin (there is one AD converter for the 16 AD;
	ADMUX |= (1 << MUX1);

	// use reference voltage 5V (Note: AVCC is AREF), manual, p281
	ADMUX |= (1 << REFS0);
}

void loop()
{
	// trigger ADC conversion
	ADCSRA |= (1 << ADSC);

	// wait until conversion is finished, see manual p285
	while (ADCSRA & (1 << ADSC));

	// read analog value, first LOW then HIGH register
	unsigned int read = ADCL + 256 * ADCH;

	double result = 5.0 * read / 1024.0;
	Serial.print(result);
	Serial.println(" Volt");

}
```

c) In die setup-Routine fügt man die folgenden beiden Kommandos hinzu, das 2. Kommando könnte ggfs. sogar weggelassen werden.

```
	// select autotrigger
	ADCSRA |= (1 << ADATE) | (1 << ADSC);

	// input source for autotrigger: free-running mode
	ADCSRB &= ~((1 << ADTS2) | (1 << ADTS1) | (1 << ADTS0));
```

Damit der Free-Running Mode auch ausgeführt werden kann, benötigt man einen einmaligen Trigger, man muss also ADSC einmal auf 1 setzen, siehe gelb markiert. In der Loop-Routine ist das aber dann nicht mehr nötig.
Die Loop-Routine lässt sich verkürzen, man muss nicht manuell eine AD Umwandlung anstoßen. Die Hardware beginnt mit der nächsten A/D Umsetzung, sobald die alte abgeschlossen ist.

```
void loop()
{
```

```
        ADCSRA |= (1 << ADSC);


        while (ADCSRA & (1 << ADSC)); -> not needed with Free Running Mode

        // read analog value, first LOW then HIGH register
        unsigned int read = ADCL + 256 * ADCH;

        double result = 5.0 * read / 1024.0;
        Serial.print(result);
        Serial.println(" Volt");
        delay(500);

}
```