Übung 07: Watchdog, Energiesparmodi, Temperatursensor

Hardware:

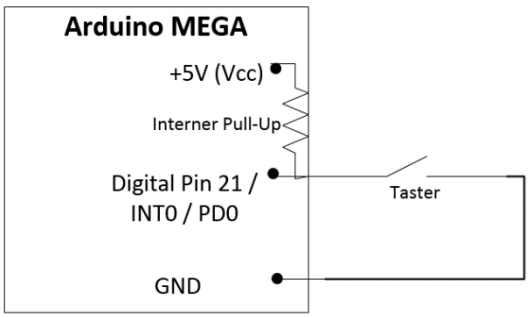
- Basis: Arduino, Steckbrett, Kabel
- Taster
- Temperatursensor TMP36


Temperatursensor TMP36

Informationen:

- Datenblatt ATmega2560: Kapitel 11 (S. 50) und Kapitel 26 (S. 268)
- Pin Mapping: https://www.arduino.cc/en/Hacking/PinMapping2560
- Interrupts: http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html

Aufgabe 1: Watchdog



Anforderung: Drückt der Benutzer nicht innerhalb von ***4 Sekunden*** den Taster an Pin `INT0` (*Digital Pin 21*, `PD0`), so soll ein ***Watchdog System Reset*** durchgeführt werden. Der Mikrocontroller soll ***neustarten***.

Hinweise:

(1) Orientieren Sie sich an der rechten Schaltung und am unten vorgegebenen Code. Dieser ist an den mit `// TODO` gekennzeichneten Stellen zu ergänzen.
(2) Datenblatt: Kapitel 12.4.2, Seite 61-63. Beachten Sie vor allem auch den C-Beispielcode. **Wichtig:**
   - `wdt_reset()` anstatt `__watchdog_reset()` verwenden!
   - `cli()` und `sei()` anstatt `__disable_interrupt()` und `__enable_interrupt()`!
(3) *System Reset Mode:* Datenblatt, S. 64
(4) Watchdog Timeout nach ***4 Sekunden***: Datenblatt, S. 66
(5) Eine Entprellung ist hier nicht gefordert.

```
#include <avr/wdt.h>

void setup() {
        Serial.begin(9600);
        Serial.println("System restart");

        // TODO: set watchdog interval to 4s and start watchdog timer

        // configure external interrupt on pin PD0
        DDRD &= ~(1 << DDD0);     // configure PD0 as input
        PORTD |= (1 << PORTD0);   // pull up, write to PORT when in INPUT mode, p68
        EIMSK |= (1 << INT0);     // turn on INT0
        EICRA |= (1 << ISC01);    // set INT0 to trigger on falling edge
        sei();                    // globally activate interrupts
}

void loop() { }                           // empty!

ISR (INT0_vect) {
       // TODO: reset watchdog within ISR
       Serial.println("ResetWDT");
}
```

Aufgabe 2: Power-Down Mode

Anforderung: ***Die Schaltung bleibt unverändert,*** nur den Code ändern wir leicht. Zu Beginn der `loop`-Methode setzen wir den Mikrocontroller in den ***Power-Down Mode***. Durch Betätigen des Tasters (externer Interrupt `INT0`) kann man den Mikrocontroller aufwecken. Die `loop`-Methode macht genau dort weiter, wo sie schlafen gelegt wurde (und fängt nicht wieder von vorne an).

Hinweise:

(1) Bereinigen Sie zunächst den Ergebniscode von 2)
- Entfernen Sie alle Watchdog-relevanten Anweisungen!
- Löschen Sie alle Anweisungen in der Interrupt Service Routine!
- Entfernen Sie alle `Serial.println()` Anweisungen

(2) Konfigurieren Sie den ***Power-Down Mode***, Datenblatt Kapitel 11, v.a. S. 54

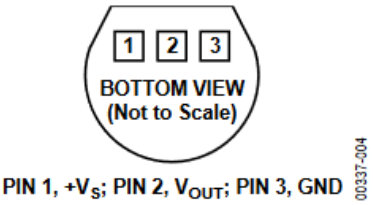
(3) Füllen Sie nun die loop-Methode mit Code:
- Legen Sie gleich zu Beginn den Mikrocontroller schlafen: Das C-Kommando **`sleep_mode()`** (Header **`avr/sleep.h`** einbinden) setzt das benötigte `SE`-Bit, Datenblatt S. 54.
- Geben Sie über die ***serielle Schnittstelle*** eine Nachricht aus, wenn Sie den Power-Down Mode betreten und verlassen. Ggfs. Delays einfügen.

(4) Der Mikrocontroller wird durch den externen Interrupt geweckt. Die bloße Existenz der ISR genügt, damit bei einem Auftreten des Interrupts der Power-Down Mode verlassen wird.

Aufgabe 3: Temperatursensor TMP36

Der Temperatursensor TMP36 liefert eine analoge Spannung, den der Mikrocontroller auswerten soll und in °C auf der seriellen Konsole anzeigen soll.

a) Der analoge Wert des Sensors soll über ***Analog Pin 2*** (ADC2 / PF2) des Mikrocontrollers eingelesen werden, siehe auch letztes Übungsblatt. Bauen Sie die nötige Schaltung auf.
Hinweis: Achten Sie auf den korrekten Anschluss des Sensors. Die rechte Abbildung zeigt die Anordnung der Pins wenn man von **unten** draufschaut. Wird der Sensor **heiß,** haben Sie Ihn falsch angeschlossen.



b) Als Basis dient der Code der letzten Übung, siehe unten. ***Vervollständigen*** Sie diesen! Beachten Sie die Vorbesprechung aus der Vorlesung und folgende Hinweise:
- Referenzspannung anpassen!
- Ergebnis in °C ausgeben!

```
void setup() {
       Serial.begin(9600);
       ADCSRA |= (1 << ADEN);                                    // enable ADC
       ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);     // ADC prescaler, p271
       ADCSRA |= (1 << ADATE)|(1 << ADSC);                       // auto-trigger
       ADCSRB &= ~((1 << ADTS2) | (1 << ADTS1) | (1 << ADTS0)); //free-running, p287
       ADMUX |= (1 << MUX1);        // select ADC2 as input pin for ADC
       // TODO: set "good" ADC reference voltage, p281
}
void loop() {
       // note: conversion is continuously triggered in free running mode
       // read analog value, first LOW then HIGH register
       unsigned int read = // TODO get converted value from ADC

       double temperature = // TODO convert integer value into temperature
       Serial.println(temperature);
       delay(1000);
}
```

c) Testen Sie, indem Sie die Temperatur erhöhen (z.B. Föhn)!