Embedded Systems
Kapitel 7: Watchdog, Energiesparmodi, analoge Eingabe

Prof. Dr. Wolfgang Mühlbauer
Fakultät für Informatik
wolfgang.muehlbauer@th-rosenheim.de

Sommersemester 2020

Inhalt

- **Watchdog Timer**
- Energiesparmodus
- Reset
- Nachtrag: Auswerten von Sensordaten

Motivation

- Was ist das Problem beim folgenden Programm?

```
uint8_t x;
x = 10;

while (x >= 0)
{
  // do something
  x--;
}
// do critical task
```

- Programmierfehler lassen sich nie ganz vermeiden!
- Gesucht: Mechanismus, der Mikrocontroller neu startet oder Interrupt auslöst, falls sich Programm "aufhängt".

Watchdog

- **Definition:**
  - Komponente, die die Funktion anderer Komponenten überwacht.
  - Erkennen von Fehlfunktion
    - „Etwas dauert zu lange“
  - Reaktion
    - Auslösen eines Interrupts oder
    - Neustart (=Reset) des Mikrocontrollers.

- **Funktionsweise**
  - Timer, der ***unabhängig*** von SW und restlicher Mikrocontroller-Hardware arbeitet.
  - Nach Aktivieren der Watchdog-Funktion wird Timer kontinuierlich in- bzw. dekrementiert
  - Wird Timer nicht rechtzeitig vor Überlauf durch SW zurückgesetzt („*Kick the dog*“)
    - Reset oder Interrupt

```
uint8_t x;

void setup() {
  start watchdog
  x = 10;
}

void loop() {
  while (x >= 0) {
    // do something
    x--;
  }
  reset watchdog
  // critical task
}
```

Aufgabe von Watchdogs

- ***Überprüfung***
  - Dauert Ausführung bestimmter Codestellen zu lange?
  - Ist SW noch aktiv und nicht bereits abgestürzt?
  - Führen HW-Probleme dazu, dass SW nicht mehr ausgeführt wird?

- ***Bei Watchdog Timeout***
  - Ziel: Überführen in wohl-definierten Zustand
  - Auslösen eines Neustarts oder Interrupts

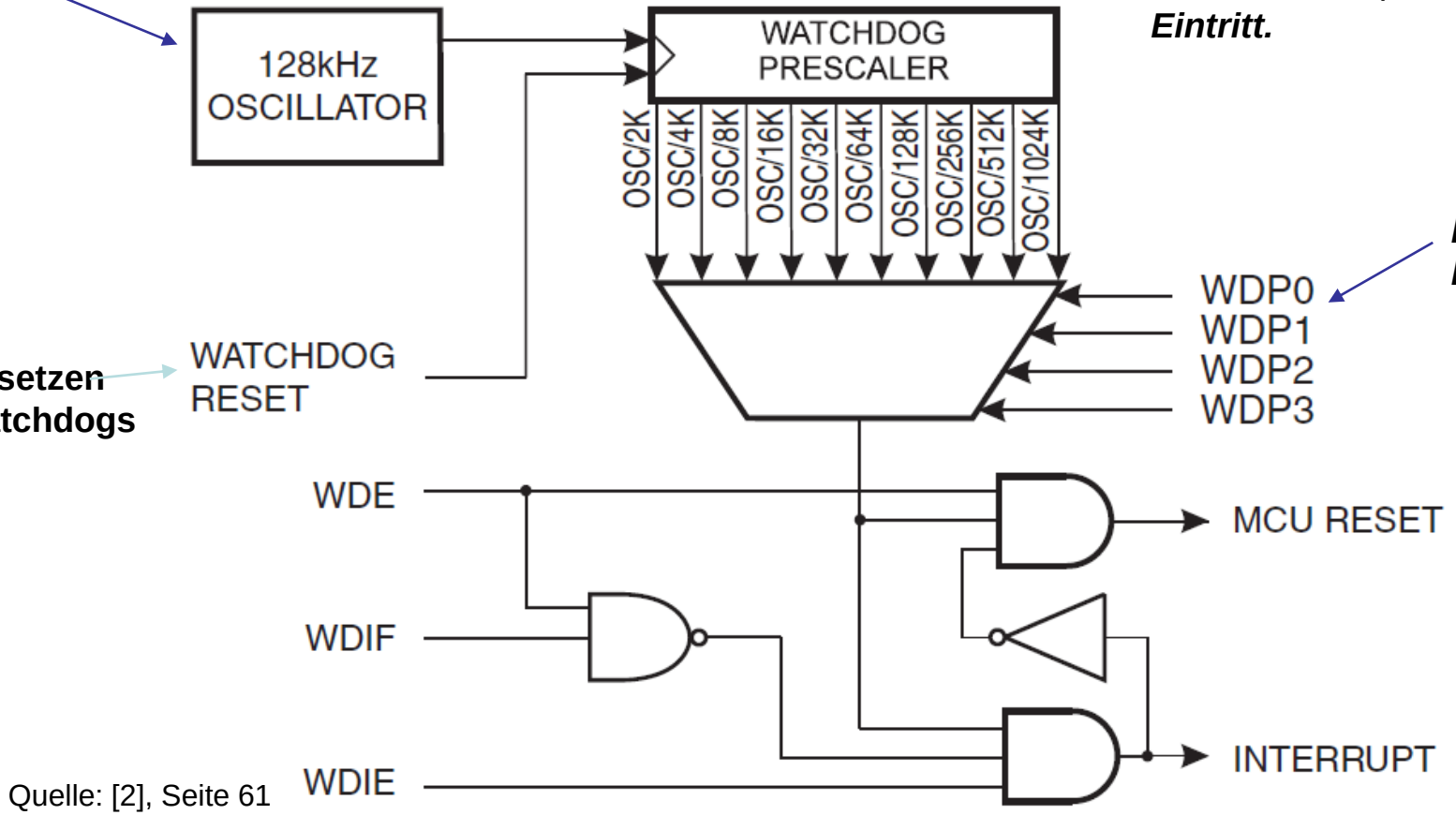
- Watchdog ***keine Allzweckwaffe*** gegen SW-Fehler!
  - Watchdog erkennt Probleme, löst sie aber nicht.
  - Probleme können nach Reset wieder auftreten.
  - Beispiel: Pathfinder-Marssonde, 1997

Watchdog beim ATmega2560

- Eigener, ***unabhängiger*** Zähler!
- Handbuch, Seite 61

Eigener Oszillator

Prescaler beeinflusst, wie lange es ohne Zurücksetzen des Watchdog Timers dauert, bis Watchdog Timeout Eintritt.



Konfiguration des Prescalers

Zurücksetzen des Watchdogs

Quelle: [2], Seite 61

Konfiguration beim ATmega2560

- Mögliche ***Reaktionen*** bei Auslaufen:
  - Nichts, Funktionalität abgeschaltet.
  - Auslösen eines Neustarts (=System Reset)
  - Auslösen eines Interrupts
  - Auslösen eines Interrupts UND eines Neustarts.

- ***2 Register***
  - `WDTCSR`: Konfiguration des Watchdogs
  - `MCUSR`: Informationen über Ursache des Resets, nach Neustart abrufbar

- ***Zurücksetzen*** des Watchdog Timers
  - Assembler-Befehl `WDT` oder `wdt_reset()` in C

- Vorkehrung, damit man Watchdog nicht versehentlich abschaltet:
  - Spezielles Vorgehen beim Beschreiben des Registers WDTCSR, siehe Handbuch Seite 61.

Inhalt

- Watchdog Timer
- **Energiesparmodus**
- Reset
- Nachtrag: Auswerten von Sensordaten

Energieverbrauch von Mikrocontrollern

- Leistungsaufnahme: Wichtiger Aspekte für viele Anwendungen.
  - Beispiel: Mikrocontroller wird an Batterie betrieben

- Verlustleistung P: $P = \boldsymbol{V_{cc}}^2 \cdot \boldsymbol{f} \cdot C$ C: Parasitäre Kapazität

- ***Reduktion des Energieverbrauchs*** durch
  - ***Verlangsamung des Systemtaktes f***
    - Leistungsaufnahme proportional zu *f*
  - ***Verringerung der Betriebsspannung*** $\boldsymbol{V_{cc}}$
    - Leistungsaufnahme proportional zu ${V_{cc}}^2$
    - Aber: Minimalspannung in der Regel notwendig
  - ***Abschalten nicht benötigter Module***
    - Verschiedene Energiespar-Modi

Exkurs: Stromverbrauch Arduino

- Stromverbrauch typischer Mikrocontroller bzw. Arduinos
  - https://www.mikrocontroller.net/articles/Leistungsaufnahme_von_Mikrocontrollern
  - https://arduino-projekte.info/stromverbrauch-arduino-wemos-boards/

- ATmega2560: Benötigter ***Versorgungsstrom***
  - Datenblatt: S. 373 / 374
  - 5,0V und 16 MHz: 21,0mA
  - 4,5V und 16 MHz: 17,5 mA
  - Nur für Mikrocontroller, nicht für Entwicklungsboard!

- Frage: Wie lange könnte man den ATmega2560 (16 MHz) mit der rechten Batterie betreiben?

4,5V, 6100 mAh
4,79 €

Energiesparmodus

- Energiespar-Modi unterscheiden sich bzgl.
  - der ***abschaltbaren Komponenten*** und
  - der **aufweckenden Ereignisse**.

- Abschaltbare Komponenten
  - Flash CPU, Oszillator, …

- Aufweckende Ereignisse
  - Externe Interrupts, Watchdog Interrupt, Speicherzugriff beendet, Timer

- Zu beachten:
  - Anderes Zeitverhalten während eines Energiesparmodus.
  - Aufwachen kann einige Zeit dauern.
  - Manche Module müssen vor Aktivieren eines Energiesparmodus ggfs. deaktiviert werden.

Energiesparmodi beim ATmega2560

	Active Clock Domains					Oscillators		Wake-up Sources						
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Osc Enabled	INT7:0 and Pin Change	TWI Address Match	Timer2	SPM/ EEPROM Ready	ADC	WDT Interrupt	Other I/O
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X
ADCNRM				X	X	X	X(2)	X(3)	X	X(2)	X	X	X	
Power-down								X(3)	X				X	
Power-save					X		X(2)	X(3)	X	X			X	
Standby(1)						X		X(3)	X				X	
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X	

Note:
1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT7:4, only level interrupt.

Quelle: [2]

Konfiguration: Energiesparmodus

- Auswahl des Energiesparmodus
  - Register `SMCR`, Handbuch, Seite 54
- Aktivieren des Energiesparmodus
  - Assembler:
    - SE-Bits im SMCR-Register muss gesetzt sein.
    - Anschließend SLEEP-Instruktion
  - C: Aufruf von `sleep_mode()` der Bibliothek `<avr/sleep.h>`
    - https://www.nongnu.org/avr-libc/user-manual/group__avr__sleep.html
- Rückkehr aus Energiesparmodus
  - Externer Interrupt: Leere Interruptroutine genügt.
  - Timer Overflow Interrupt
  - Watchdog Interrupts

Inhalt

- Watchdog Timer
- Energiesparmodus
- **Reset**
- Nachtrag: Auswerten von Sensordaten

Reset

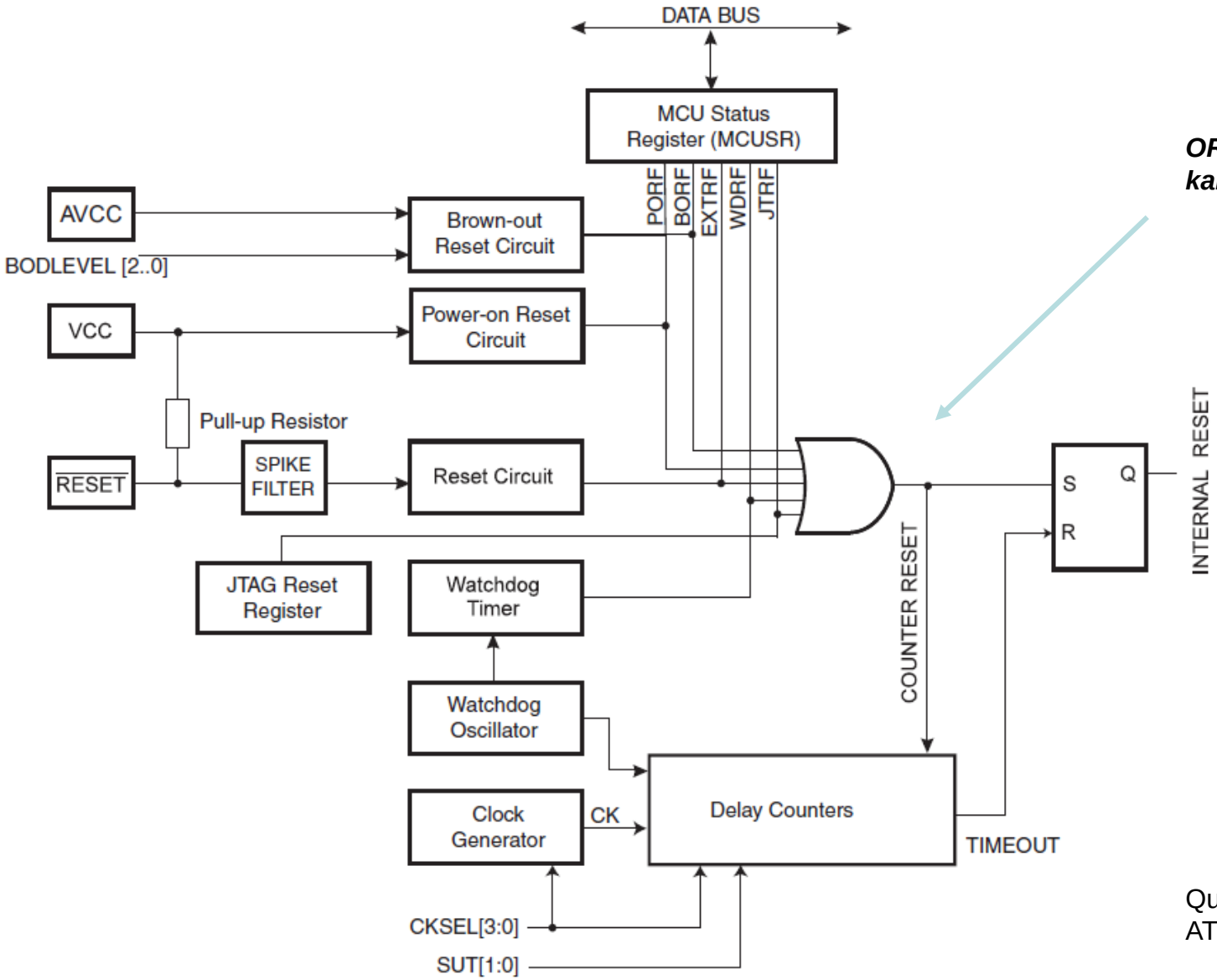
- **Definition**
  - *Asynchrones Ereignis, das ein eingebettetes System dazu veranlasst, die CPU und die meisten Komponenten eines Mikrocontrollers von einem wohldefinierten, bekannten Zustand zu starten.*

- **Bei Reset**
  - Initialisieren aller Register und I/O Ports auf Default-Werte
  - Künstliches Delay → Spannungswerte sollen sich stabilisieren
  - Ausführen der ersten Instruktion an der Adresse 0x0000, wo üblicherweise ein JMP zur Reset-Routine abgelegt ist.
  - Reset-Routine initialisiert Stack Pointer etc. und enthält als letzte Anweisung Sprung auf Main-Routine (Arduino Sketch: `setup`).

Arten von Resets

- ***Power-On Reset***
  - Auslösung falls Versorgungsspannung einen bestimmten Schwellwert *überschreitet*.
- ***Brown-Out Reset***
  - Auslösung falls während des Betriebs die Versorgungsspannung unter einen bestimmten Wert *fällt*.
  - Oft abschaltbar, Schwellwert teils konfigurierbar.
- ***External Reset***
  - Auslösung Eingangspin (RESET) auf LOW gezogen wird.
  - Möglich durch Drücken des roten Tasters auf Arduino-Board.
- ***Watchdog Reset***
- ***Internal Reset***
  - Reset kann durch SW Instruktion in Mikrocontroller-Programms ausgelöst werden.

Resets beim ATmega2560



OR-Gatter: Jeder der Quellen kann auslösen.

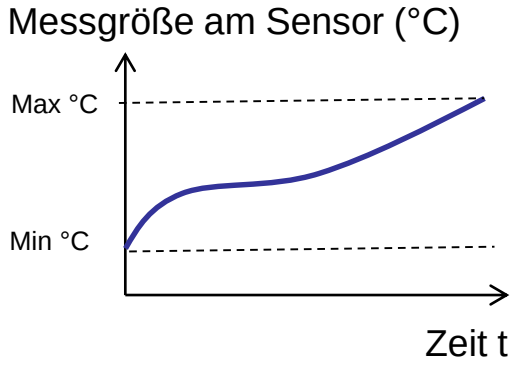
Quelle: Handbuch ATmega2560

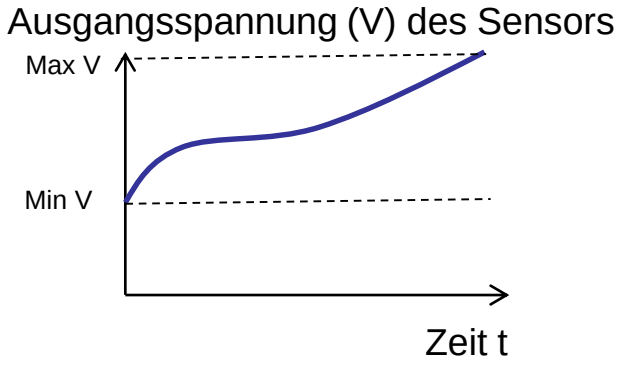
Inhalt

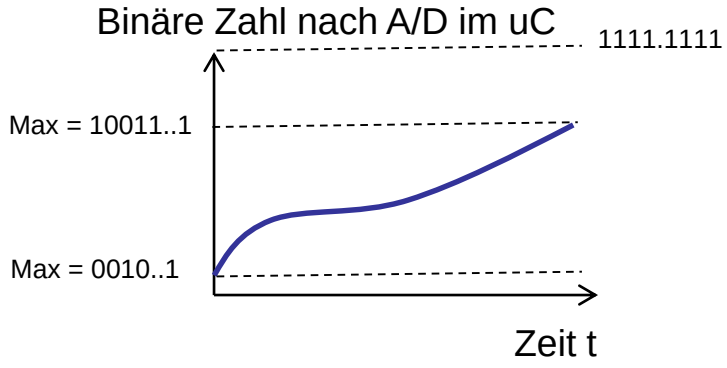
- Watchdog Timer
- Energiesparmodus
- Reset
- **Nachtrag: Auswerten von Sensordaten**

Nachtrag: Auswerten von Sensordaten

- Fast jeder Sensor liefert *Ausgangsspannung* bzw. ändert Widerstand in Abhängigkeit der *Messgröß*e (°C, Feuchtigkeit, Helligkeit, …).
- Meist **linearer** Zusammenhang zwischen *Ausgangsspannung* und *Messgröß*e.
- A/D Wandler in Mikrocontroller wandelt *Ausgangsspannung* in *binäre Zahl* um.
- Wie schließt man dann im Mikrocontroller-Programm von binärer Zahl auf Messgröße?





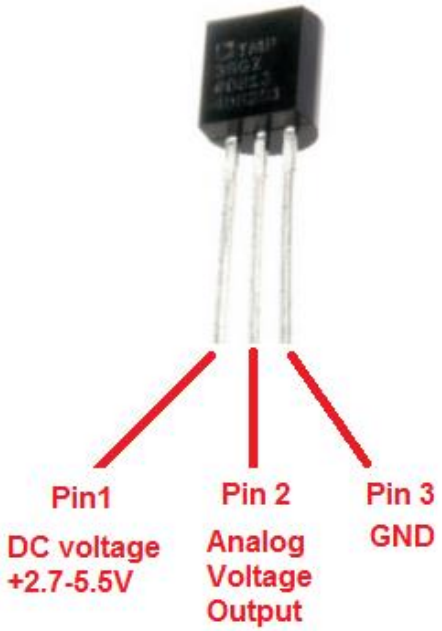


Beispiel: Sensor TMP36 von Analog Devices

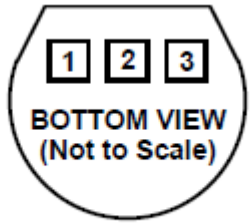
- Temperatursensor
  - Ausgangsspannung proportional zur Umgebungstemperatur in °C
  - http://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf

- Beschaltung beachten!
  - Sicht von unten
  - **Wenn er heißt wird, umstecken!**

- Ziel: Verdeutlichung wie man Sensordaten innerhalb eines Mikrocontroller-Programms auswertet.



Quelle: http://www.learningaboutelectronics.com/images/TMP36-pinout.png (abgerufen am 17.05.2016)



PIN 1, $+V_S$; PIN 2, V_{OUT}; PIN 3, GND

Quelle: Datenblatt TMP36

Schritt 1: Wertebereich der Ausgangsspannung

- **Arbeitsbereich** des Sensor
  - In welchem Bereich besteht ein **linearer** Zusammenhang zwischen *Messgröße* und *Ausgangsspannung*?
  - In welchem Bereich ist die *Abweichung* hinreichend klein?
  - Ist dieser Bereich im Einklang mit der Anwendung?
  - Beispiel TMP36: -40°C bis 125°C (Abweichung ist max. 1°C)

- Was ist bei diesen Randbedingungen die ***minimale*** und ***maximale Ausgangsspannung*** des Sensor?
  - TMP36: 750 mV bei 25°C, Output Scale Factor von 10 mV/°C
  - Minimale Ausgangsspannung:
    - 750 mV + 10 mV/°C * (125°C – 25°) = 1750 mV = 1,75 V
  - Maximale Ausgangsspannung:
    - 750 mV + 10mV/°C * (-40°C -25°) = 100 mV = 0,1 V

Schritt 2: Wertebereich der binären Zahl

- Wahl einer geeigneten ***Referenzspannung***
  - Maximaler Wert der Ausgangsspannung sollte möglichst knapp unter Referenzspannung liegen, um gute Genauigkeit zu erzielen.
  - Welche Referenzspannungen gibt es beim ATmega2560?
  - Hier: Referenzspannung, S. 281 2,56 V

- ***Maximum und Minimum der binären Zahl*** (=Ergebnis des A/D Wandlers), S. 280
  - Maximum: 1,750 V * 1024 / 2,56 V = 700
  - Minimum: 0,100V * 1024 / 2,56 V = 40

Schritt 3: Umrechnung in Messgröße

	Minimum	Maximum
Messgröße	-40°C	125°C
Ausgangsspannung	100mV	1750mV
"Binäre" Zahl	40	700

Annahme: Referenzspannung 2,56V

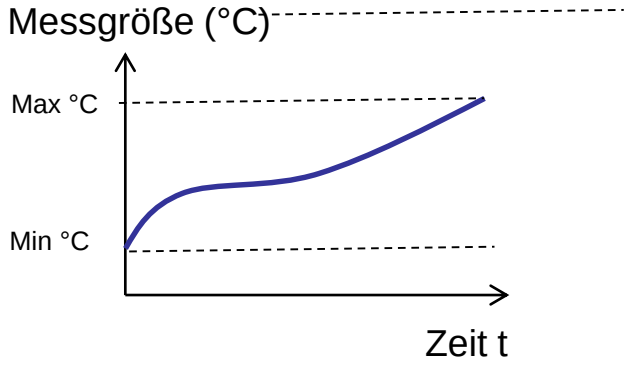
- ***Gesucht:***
  - Formel, die zu binärer Zahl direkt die entsprechende Messgröße liefert.
  - Zwischen binärer Zahl und Messgröße besteht auch linearer Zusammenhang!

- ***Ansatz***: $y = mx + t$
  - y: Messgröße, hier °C
  - x: binäre Zahl, in Mikrocontrollerprogramm verfügbar
  - m: "Steigung" der Geraden, t: "Achsenabschnitt"
  - *2* Punkte gegeben, m und t bestimmen!

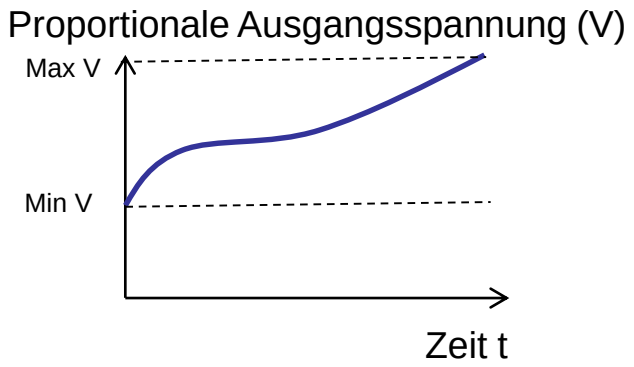
- ***Lösung***: y=0,25x – 50

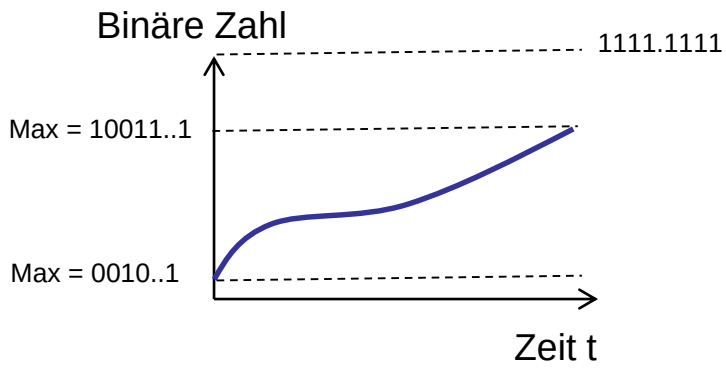
Auswerten von Sensordaten

- Allgemeingültiges Rezept zum Auslesen von Sensordaten



- ***Arduino-Hilfsfunktion***
  - `map(value, fromLow, fromHigh, toLow, toHigh)`
  - https://www.arduino.cc/en/Reference/Map
  - Aber auch hier muss man sich die Werte `fromLow, …` `toHigh` selbst überlegen!





Quellenverzeichnis

[1] G. Gridling und B. Weiss. *Introduction to Microcontrollers*, Version 1.4, 26. Februar 2007, Kapitel 2.7, verfügbar online: https://ti.tuwien.ac.at/ecs/teaching/courses/mclu/theory-material/Microcontroller.pdf (abgerufen am 08.03.2017)

[2] Datenblatt ATmega2560, http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf, (abgerufen am 19.03.2017)

[3] https://www5.in.tum.de/lehre/seminare/semsoft/unterlagen_02/marssojourner/website/mars/zusammen.htm (abgerufen am 08.05.2017)

[4] https://de.wikipedia.org/wiki/Priorit%C3%A4tsinversion (abgerufen am 08.05.2017)