Berufliches Gymnasium
am BSZ Bau und Technik Dresden
am BSZ Elektrotechnik Dresden

Hochschule für Technik und Wirtschaft Dresden

Besondere Lernleistung

Kurs BG07
Leistungskurs Datenverarbeitung

ClassPad 300 PLUS Anwendungen im Chemieunterricht

von

Johannes Höntsch

BG07 L3DV1

Dresden, 22.02.2010

Inhaltsverzeichnis

I. Vorwort

Seit dem Schuljahr 2007/2008 ist der „CASIO ClassPad“ als grafikfähiger programmierbarer Taschenrechner mit Computer Algebra System (CAS) auch an Beruflichen Gymnasien in Sachsen zugelassen. Er erlaubt neben symbolgetreuer Darstellung mathematischer Funktionen und Operationen Tabellenkalkulation, geometrische Zeichnungen, Kalkulation mit statistischen Daten, 3D-Plots sowie Kommunikation mit anderen Taschenrechnern, um nur einige Funktionen zu nennen.

Diese Besondere Lernleistung soll auf Basis der Belegarbeit von Martin Anhut und Johannes Höntsch aus dem Jahr 2008/2009 die Verbindung von Mikrocontroller und ClassPad via UART vertiefen. Ziel dieser Arbeit ist es, auf Grundlage der bisherigen Erkenntnisse die Messdaten eines Temperaturfühlers zum Zwecke chemischer Experimente am Taschenrechner auszugeben.

Besonderen Dank möchte ich an Professor Paditz, Professor Kühn und Frau Riester sowie Herrn Große richten, welche dieses Projekt betreut und unterstützt haben. Ebenso Dank an Dr. Thomas Hoffmann für kleine Tipps und Bereitstellung von Literatur als auch Bauteilen.
Des Weiteren möchte ich mich für das Interview sowie Umfragen und Tests bei den Lehrern des BSZ Bau und Technik Herrn Dr. Schulz (Mathematik) und Frau Krug (Chemie) bedanken.
Weiterhin Dank an Herrn Wagner (ET-Lehrer am BSZ Bau und Technik) für die Bereitstellung professioneller Geräte zur Platinenherstellung.
Dank gebührt auch den Schülern Carl Richter und André Schubert (BG07 BSZ Bau und Technik), Dirk Bindszus (WGy07 BSZ für Wirtschaft Pirna) und Alrik Künne (BSZ Elektrotechnik), die maßgeblich zur Verbesserung der Komponenten im Rahmen der Evaluierung beigetragen haben.

Hinweise auf Dokumente im Anhang sind mit „▷ Anlagenummer“ gekennzeichnet.
Alle verwendeten Programme sind nach Anwendungsgebiet im Anhang (▷ S01) aufgelistet.

II. Zielsetzung und Zweck der Arbeit

Primär soll das zu entwickelnde Messwerterfassungsgerät zu Messungen bei Schülerexperimenten in Chemie genutzt werden. Auch bei Lehrerdemonstrationen ist der Einsatz möglich. Es wird einen Temperatursensor besitzen und modular erweiterbar sein. So können thermodynamische Reaktionen protokolliert oder in Kombination mit mehreren Sensoren die daraus resultierenden Zusammenhänge deutlich werden. Die Auswahl der Experimente ist hier groß, da sie immer einen festen Bestandteil des Chemieunterrichts darstellen. Exemplarisch ist hier die in Klasse 12 durchzuführende energetische Betrachtung chemischer Reaktionen. Sie ist ohne Temperaturerfassung oder -beobachtung nicht möglich. Ein Beispiel ist die experimentelle Ermittlung der Reaktionsentalpie und sich daraus ergebende Rechnungen, wie die freie Entalpie nach *Gibbs-Helmholz* oder die Entwicklung eines Grundverständnisses der Bedeutung von exothermen und endothermen Reaktionen sowie den damit verbundenen Eigenschaften; den Begriffen Wärme, innere Energie, Entalpie und Entropie, was auch im Einklang mit dem Lehrplan für das Fach Chemie steht (vgl. [L1], S. 20 ff.).

Natürlich beschränkt sich die Erfassung von Temperatur (und je nach Erweiterung auch anderer Größen) nicht nur auf Chemie, sondern spielt auch in den Fächern Physik und Biologie eine Rolle. Die Ausrichtung dieser Arbeit in Anwendungsbeispielen ist jedoch chemischer Natur.

Die Übertragung der gewonnenen Daten erfolgt über die Funkstrecke, welche bereits in der Belegarbeit dokumentiert wurde. Zu Diagnosezwecken wird auch ein Anschluss über Kabel möglich sein. Die Ausgabe der Ergebnisse erfolgt am ClassPad. Hauptaugenmerk liegt hier auf der Anzeige der aktuellen Daten und dem Verlauf der Messwerte in Form von Diagrammen und Tabellen. Nicht vorgesehen sind komplette Auswertungen nach energetischen Gesichtspunkten. Um aber die Integration des Taschenrechners in andere Fächer zu fördern, sind Regressionen der Daten vorgesehen.

III. Organisation

Der Projektplan ▹ A01 gibt eine grobe Übersicht über die notwendigen Schritte bzw. die Planung des Projektes. Probleme bei der Erarbeitung der Zielsetzung führten allerdings zu einem sich immer in Änderung befindlichen Plan. Die wesentlichen Punkte sind rot markiert (Meilensteine). Eine Mindmap ▹ A02 zeigt nach verschiedenen Kriterien aufgeschlüsselt die grundlegenden Teile der Besonderen Lernleistung.

Im Wesentlichen gliederte sich die Arbeit in die drei Arbeitsschritte 1. der Recherche, 2. des Experimentierens und 3. Dokumentierens, wobei die ersten beiden Begriffe auch vertauschbar sind. Die zwei Hauptgebiete der Temperaturerfassung (Mikrocontrollerprogrammierung, Schaltung) und Erstellung eines Auswertungsprogramms für den ClassPad wurden parallel durchgeführt. In den beiden Bereichen gab es ebenfalls eine parallele Arbeitsweise, welche aber nicht immer möglich war und somit teilweise zur chronologischen gewechselt wurde.

IV. CAS – Einsatz im Unterricht und technische Daten

Die Abkürzung CAS steht für Computer Algebra System und bezeichnet technische Verarbeitungsanlagen, die mathematische Probleme nicht in erster Linie nummerisch, sondern mittels Computeralgebra lösen. Der ClassPad ist nach dem CASIO CFX 9970 G und dem Algebra FX 2.0 die dritte Serie von CASIO (vgl. [K1], [K4]), die über diese Funktionalität verfügt. Der CFX 9970 G ist am wenigsten verbreitet. Dessen Nachfolger, der Algebra FX 2.0, besitzt nicht mehr das teure Farbdisplay, dafür aber ein erweitertes CAS.

Seit der Serie ClassPad ist die Bedienung über Touchscreen möglich, was die Handhabung wesentlich vereinfacht. Der Vorgänger des 300 PLUS hat ein kontrastärmeres Display und keinen Mini-USB Port. Zur Zeit gibt es bereits den Nachfolger ClassPad 330, der im Vergleich zum ClassPad 300 die Softwareversion 3.03 bereits beinhaltet. Bis auf die unterschiedlich vorinstallierten Betriebssystemversionen gibt es keine Unterschiede. Eine Aktualisierung des installierten OS ist problemlos möglich.

Die Nutzung in Klassenstufe 11 zur Einführung beschränkt sich im wesentlichen auf den Mathematikunterricht, in dem die Grundfunktionen wie Terme umformen, Gleichungen lösen, Funktionen zeichnen und Grundlagen der Stochastik behandelt werden. Somit wird auch in die

Handhabung von computerähnlichen Taschenrechnern eingeführt. Bereits hier ist die integrierte Programmentwicklung nützlich, damit die Algorithmen zu Berechnungen nicht immer erneut eingegeben werden müssen.

Ab Klasse 12 ist der Taschenrechner ein unabdingbares Werkzeug, da mit ihm auch komplizierte Integrale und Differenziale in kurzer Zeit berechnet werden können, sofern eine Lösung existiert.

In der Jahrgangsstufe 13 sind vor allem die Darstellungsmöglichkeiten bei diesem Gerät interessant. Vektoren können beispielsweise symbolgetreu aus der Aufgabe übernommen werden. Dies trifft auch auf andere Ausdrücke wie Brüche, Potenzen, Summen, Integrale, Logarithmen und vieles weitere zu.

Da mathematische Berechnungen auch in anderen Fächern eine Rolle spielen, sind die Funktionalitäten nicht nur auf Mathematik beschränkt. Gerade für die Stöchiometrie (chemisches Rechnen) ist der Taschenrechner ein wichtiges Instrument. Bisher beschränkte sich aber dessen Einsatz nur auf das Lösen von Verhältnis- oder logarithmischen Gleichungen. Das Ergebnis dieser Arbeit soll die Integration des ClassPads in den (Chemie-) Unterricht fördern. So kann er nicht nur zur manuellen Auswertung in Rechnungen, sondern auch zum Erfassen der Daten genutzt werden.

Die Vorteile bergen jedoch auch Gefahren in sich, denn

> „die Arbeit mit dem CAS ist vor allem hilfreich für leistungsstarke Schülerinnen und Schüler, die mathematisches Verständnis haben und denen der Rechner als sinnvolles Hilfsmittel wirklich Erleichterung bringt, [… jedoch] kann das mathematische Verständnis leistungsschwacher auf der Strecke bleiben.“ (vgl. [K2]).

Dies ist besonders bei Problemen mit Syntax und Ergebnisinterpretation der Fall. Daher gilt der Grundsatz, dass der Taschenrechner kein Ersatz für Denkleistung ist. Der mathematische Ansatz muss im Kopf bereits bestehen, ehe das Gerät genutzt werden kann (vgl. [K2]). Hier ist auch immer abzuwägen, ob der Einsatz sinnvoll ist, denn Tippfehler und deren anschließende Suche sind zum Teil zeitraubender als eine Zwischenrechnung auf dem Papier.

Der Rechner besitzt zwei Kommunikationsports: einen vier-poligen USB-Mini-Port und einen drei-poligen Anschluss. Letzterer wurde bereits in der Belegarbeit genutzt und wird daher auch hier in Gebrauch sein. Grund ist die leichte Ansteuerung seitens der ClassPad-Firmware wie auch aus Sicht der Mikrocontroller. Die USB-Schnittstelle wird für den Anschluss und Datenaustausch an einem Computer benötigt. Ob sie auch für den Datenaustausch zwischen Taschenrechner und Datenerfassungseinheit über die Funkmodule geeignet wäre, ist aus oben genannten Gründen nicht Teil der Arbeit.

Die Datenübertragung der seriellen Schnittstelle erfolgt mit 9,6 kBd, 38,4 kBd oder 115,2 kBd. Der Prozessor vom Typ SH7291 ist ein 32-Bit RISC Mikrocomputer von Renesas Technology mit einer Taktfrequenz um 32 MHz (vgl. [L7], #7 ff.). Der Programmspeicher mit 5,4 MB bietet einigen Platz für Programme, Spiele und Add-Ins. Für Rechnungen stehen knapp 503 KB RAM zur Verfügung (vgl. [Q6], S. 725 – „5 Technische Daten“).

V. Analyse zu Hard- und Software des Projektes

Dieses Kapital beschreibt kurz die grundlegenden Voraussetzungen in Hard- und Software und führt auf Basis der Zielstellung der Arbeit eine IST- und SOLL-Analyse durch. Es gibt auch einen groben Überblick über die entsprechenden Arbeitsschritte.

V. 1 IST-Analyse

Vorhanden sind bereits die Funkmodule (Empfang und Senden) sowie Mikrocontroller mit entsprechendem Steuercode, ebenso die Ansteuerung der seriellen Schnittstelle via UART. Die AVR Programmierung in C war auch Bestandteil der Belegarbeit, auf deren Basis dieses Projekt entsteht. Daraus resultierende Programmier- und Diagnosehardware ist gleichermaßen vorhanden wie Grundlagen und Software für die Erstellung der Hardware (Schaltpläne, Platinenlayout, Ätzhemikalien).

Das von CASIO angebotene Messinstrument EA-200, mit dem ebenfalls Temperaturmessungen möglich sind, verfügt nicht über eine Funkverbindung. In entsprechenden Tests wird das entwickelte Produkt mit dem EA-200 verglichen.

V. 2 SOLL-Analyse

Nötig wird Sensorik für die Erfassung der physikalischen Größen. Um die Temperatur zu messen, soll ein preisgünstiger Silizium-Fühler vom Typ KTY genutzt werden. Die Messwerterfassung erfolgt mit dem Analog/Digital-Wandler des Mikrocontrollers Atmega168 (▷ A03). Dann wird der Wert mittels UART an den Funkcontroller weitergegeben, der ihn über die RFM12-Funkmodule ausgibt. Vom Empfänger werden die Daten direkt an den ClassPad übergeben und dort angezeigt.

Die Messung soll dabei einfach oder im Diagramm mit variabler Zeit und veränderlichen Messabstand möglich sein. Dazu muss der AVR auch ClassPad-Zeichen empfangen und verarbeiten können. Für diese Zwecke sind zwei unabhängige Funktionen zu schreiben und in das Hauptprogramm, das die Messung vornimmt, zu integrieren.

Der Nutzer soll die Möglichkeit haben, die Werte in den technischen Grenzen frei zu wählen. Dazu zählt auch die Berechnung der Temperatur im ClassPad (→ hohe Transparenz). Für die Anwendung ist eine Bedienungsanleitung anzufertigen und mit dieser die Evaluierung durchzuführen.

VI. Temperatur erfassen und berechnen

Um die Temperatur zu erfassen, ist ein Sensor nötig. Es gibt vom einfachen Heiß- oder Kaltleiter aus Silizium bis hin zum voll digitalem I²C bzw. TWI fähigen Modul je nach Anforderungen und Geldbeutel eine breite Palette an Sensoren. Für dieses Projekt ist der Temperatursensor vom Typ KTY (Infineon) mit einem annähernd linearen variablen Widerstand zwischen -30 °C und +130 °C vom Preis-/Leistungsverhältnis her ideal. Um den sich in Abhängigkeit der Temperatur ändernden Widerstand per Mikrocontroller zu erfassen, muss eine A/D-Wandlung erfolgen. Die folgenden Kapitel beschreiben die Wandlung, Widerstands- und damit Temperaturmessung. Einen ersten Test zur Ermittlung des Widerstandes ohne Mikrocontroller stellt dazu das Protokoll I (▷ PR01) dar.

VI. 1 Der Analog/Digital Wandler am AVR

Es gibt prinzipiell zwei Methoden, um ein analoges Signal in einen binären String zu überführen. Eines ist das Parallelverfahren, das bauteilaufwändig und daher schnell arbeitet. Beim zweitem, dem Wägeverfahren, wird mittels D/A-Wandler und Komparator geprüft, ob der Wert einer Referenzspannung entspricht oder nicht. Den geringen Bauteilaufwand kompensieren viele Taktzyklen (vgl. [L2]). Der AVR arbeitet nach dem zuletzt umrissenen Verfahren (vgl. [Q1], S.244 – Figure 21-1.).

Der genutzte Mikrocontroller vom Typ ATmega168 besitzt zwei wesentliche Initialisierungsregister und ein Datenregister sowie ein Register für Interruptsteuerung oder externe Trigger, welche hier aber nicht von Bedeutung sind. Die Auflösung beträgt 10 Bit.

Zuerst wird der Eingang in *ADMUX* festgelegt, von dem der ADC die Signale wandeln soll. Die Eingänge beim ATmega168 sind PC0 bis PC5 (entsprechend ADC0 bis ADC5). Das untere Nibble in *ADMUX* ist für diese Kanalwahl reserviert. Die zwei höchsten Bits (6 und 7) des achtstelligen Registers repräsentieren die Referenzspannung. Es gibt drei Möglichkeiten diese zu wählen: eine externe, die allerdings V_{CC} nicht überschreiten darf, V_{CC} als V_{REF} oder eine interne 1,1 V-Referenzspannung. Hier fiel die Wahl auf V_{CC} als Vergleichsspannung. So wird zu Lasten der Quantisierung vermieden, dass ein zu hohes Signal an ADCx anliegt. Quantisierung ist die Einteilung eines analogen Spannungsbereiches in Abhängigkeit von einer Referenzspannung, um den binären Wertigkeiten analoge Wertebereiche zuzuweisen. Je feiner diese Stufen sind, desto näher ist das binäre Signal am analogen: $\frac{5\,V}{2^{10}} = 4{,}88\,mV$ im Vergleich zu $\frac{1{,}1\,V}{2^{10}} = 1{,}07\,mV$. Mit 5 V als Referenzspannung sind die Quantisierungsstufen also gröber als bei 1,1 V.
Um die Modularisierung durch die Anschlussmöglichkeit weiterer Komponenten zu realisieren, wird mit 5 V gearbeitet werden. Damit diese gewährleistet sind, können 4 Batterien á 1,5 V mit einer 5,1 V-Zener-Diode oder einem Spannungsregler wie dem LM 336 genutzt werden. Jedoch ist zu beachten, dass die Komponenten selbst auch eine Betriebspannung haben bzw. verbrauchen.

Das zweite Register, das wichtig für die Initialisierung ist, ist das *ADCSRA*. Über Bit sieben wird der ADC erst aktiviert und Bit sechs startet eine Umwandlung. An den letzten drei Bits stellt man einen internen CPU-Teiler ein, der die Abtastrate festlegt. Diese sollte zwischen 50 kHz und 200 kHz liegen (s. [Q1], S.246 – 21.4). Bei einer CPU-Frequenz von 8 MHz liegt der Teiler somit zwischen 40 und 160.

Ist eine Wandlung beendet, wird das Ergebnis in den Datenregistern *ADCL* und *ADCH* abgelegt. Angesprochen werden sie entweder einzeln, wobei ADCH dabei vorher um acht Bits nach links geschoben werden muss, oder über *ADCW*, was beim ATmeag168 möglich ist. Eine Übersicht darüber gibt ▹ A04.

VI. 2 Digitalisierten Spannungswert ausgeben

Im vorherigen Kapitel wurden die essentiellen ADC-Register erläutert. Nun muss der in *ADCW* befindliche Spannungswert ausgegeben werden. Das ist für den Anfang am einfachsten mittels UART zu lösen. ▹ PAP01 ist der zugehörige Programmablaufplan, der folgend kurz erklärt wird

(der Quellcode ist unter ▹ QC01 angehängt).
Erst werden die entsprechenden Bibliotheken und einige Funktionen eingebunden bzw. definiert (1). Aus Diagnosegründen erfolgt nach einer Wartezeit von einer Zehntelsekunde für die Initialisierung des UART (2) die Ausgabe des Zeichenstrings „UART OK“ (3). Jetzt werden die Register gesetzt (4) und es folgt eine Probe-Umwandlung, um den ADC „aufzuwärmen“ (5). Das Ergebnis des Vorganges muss gelesen werden, wird aber nach Beginn der Messschleife (6) wieder verworfen. Nun beginnt die eigentliche Erfassung der Messwerte. Dazu wird das ADCW-Register 16 Mal hintereinander gelesen und die Summe der Werte gebildet (7). Im Anschluss wird durch das rechtsseitige Schieben um vier Bit eine „mikrocontrollerfreundliche“ Division von $2^4 = 16$ vorgenommen (8). Danach muss die Umwandlung der erzeugten Ganzzahl in eine Zeichenkette erfolgen, damit sie über *uart_puts();* ausgegeben werden kann (9).

Die TTL-Signale der UART-Schnittstelle werden mit einem Wandler in den Signalpegel nach RS232-Norm überführt und mittels HTerm von COM1 gelesen (vgl. [L4], Kapitel III., S. 4 f.).

Zur Diagnose ist Punkt (3) eingebaut. Im fertigen Produkt spielen solche Funktionsaufrufe keine Rolle mehr. Sollte dennoch die Fehlerverfolgung anhand dieser Befehle durchgeführt werden müssen, lassen sie sich leicht wieder in den Mikrocontroller implementieren.

VI. 3 Widerstandsmessung mittels AVR

Mit allen Mikrocontrollern ist der Widerstand eines Bauelements nur indirekt ermittelbar. Als einzigen Wert kann der Analog/Digital-Wandler eine elektrische Spannung messen. Ist, wie in Kapitel VI. 2 beschrieben, die A/D-Wandlung am AVR nutzbar, so gibt der Computer einen Wert aus, der sich auf die eingestellte Referenzspannung bezieht.

Der Wandler arbeitet mit 10-Bit Auflösung. Das heißt, er kann im Bereich von 2^0 bis 2^9 die gemessenen Spannungen ausgeben. Daraus folgt ein dezimaler Bereich von 0 bis 1023 (= 1024 Werte). Das Datenblatt gibt für die Berechnung der Spannung aus dem ADC-Wert die Formel

$$U_1 = \frac{U_{REF} \cdot ADCW}{1024}$$ an (vgl. [Q1], S. 254 – 21.7).

Mit dieser errechneten Spannung wird nun der Widerstand hergeleitet. Durch die Verwendung eines

Spannungsteilers kann die Spannungsteilerregel angewandt werden. Die Verschaltung beschreibt ▷ SL01 im Anhang. Es ist eine sehr einfache wie auch zweckmäßige Anordnung.

Eine ähnliche Möglichkeit, Widerstände zu ermitteln, bietet die Wheatstone-Brücke. Dabei handelt es sich im Grunde um zwei parallele Spannungsteiler. Gemessen wird dabei der Potentialunterschied zwischen den Spannungsteilern. Diese Schaltung ist genauer, was gerade die minimalen Widerstandsänderungen von Temperatursensoren angeht (vgl. [L3], S. 147 – 1.2). Jedoch ist der Bauteilaufwand und der des Ausgleichs der Brücke höher als bei dem vorher genannten Messverfahren. Zudem ist die Genauigkeit eines einfachen Spannungsteilers ebenfalls ausreichend.
Alternativ kann mit einer Konstantstromquelle, bestehend aus einem Transistor und drei Widerständen, durch das Prinzip der Gegenkopplung ein konstanter Strom bereitgestellt werden, um mit dem Ohmschen Gesetz den Widerstand zu ermitteln. Allerdings kann hier der Bezug zur Masse Probleme bereiten (vgl. [L12], S. 4 ff.).

Allgemein gilt für Spannungsteiler, dass das Verhältnis einer Teil- zur Gesamtspannung gleich dem von Teil- zum Gesamtwiderstand ist:
$\frac{U_1}{U_{ges}} = \frac{R_1}{R_1 + R_X}$. Umgestellt nach R_X ergibt sich also $R_X = \frac{R_1 \cdot U_{ges}}{U_1} - R_1$, wobei U_1 die durch den Mikrocontroller gemessene Spannung darstellt. Da sowohl der AVR als auch der Sensor später die gleiche Spannungsquelle besitzen, kann $U_{REF} = U_{ges}$ gesetzt werden. Beim ineinander einsetzen der Formeln fällt die Gesamtspannung heraus, ist also nicht mehr von Bedeutung:
$R_X = \frac{R_1 \cdot 1024}{ADCW} - R_1$.
Bis auf R_1 sind nun alle Größen bekannt beziehungsweise messbar. Der Wert von R_1 sollte in etwa dem von R_X entsprechen. Für variable Widerstände wird der Mittelwert aus dem Minimum und Maximum empfohlen. Bei einem minimalem Widerstand von 1230 Ω (-30 °C) und einem maximalen von 4144 Ω (+130 °C) ergibt sich daher ein sogenannter Shunt von 2687 Ω. Für kleinere oder größere Messwiderstände wird der nutzbare Messbereich nur kleiner: ein 1 kΩ-Shunt hat nur noch 260 von 1023 möglichen Werten im zulässigen Bereich, mit einem Widerstand von 3 kΩ sind nur noch 29 % der 10-Bit Auflösung nutzbar.

Eine andere Möglichkeit, den Messwiderstand zu ermitteln, ist, ihn mathematisch exakt zu berechnen. Dazu sind zwei Funktionen nötig: eine, die den geringsten und eine, die den höchsten

Wert darstellt. Über das Datenblatt ergeben sich so: $R_{min} = \frac{x \cdot 1024}{x + 4144}$ und $R_{max} = \frac{x \cdot 1024}{x + 1230}$. Die Variable x kennzeichnet dabei den Shunt. Nun soll zwischen Minimum und Maximum ein größtmöglicher Abstand entstehen: $R_{max} - R_{min} \rightarrow max$. Wird das Differential nach x gleich Null gesetzt und gelöst, dann ergibt sich 2257,68 als idealer Wert. An dem Verlauf des Graphen im Diagramm 1 wird der erläuterte Zusammenhang deutlich.

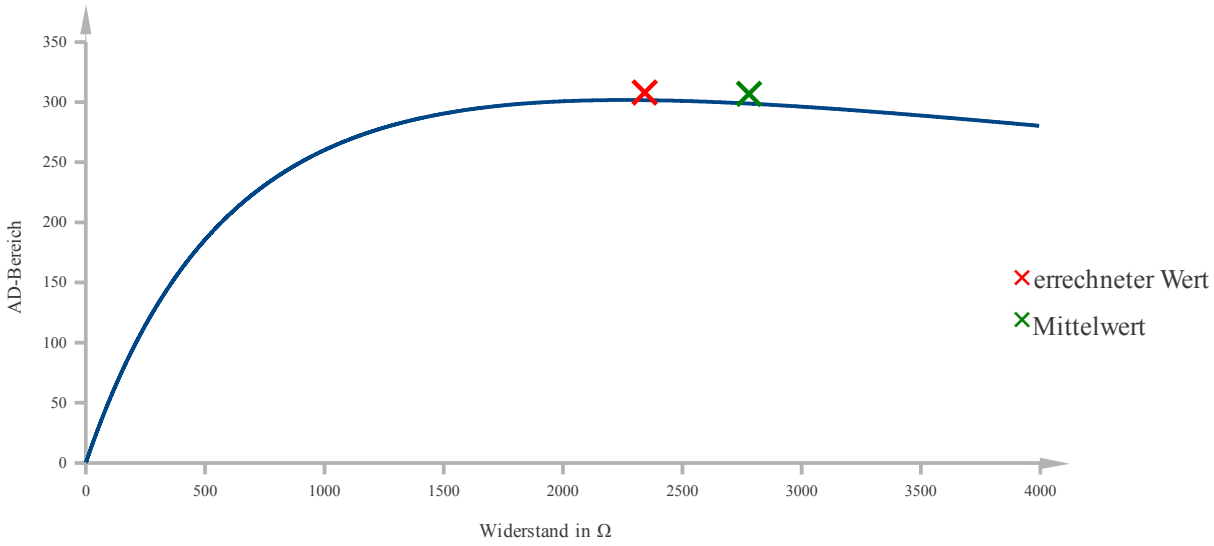


Diagramm 1: gültiger AD-Bereich in Abhängigkeit das Shunt-Wertes

Vergleicht man den zulässigen Bereich der Mittelwertrechnung und den der Extremwertaufgabe, so liegt der der EWA um eine Stufe höher. Dies ist zwar kein praktikabler Gewinn, zeigt aber, dass der Mittelwert nur eine Näherung ist.
Ebenso wird deutlich, dass große Widerstände den Bereich weniger stark beeinflussen als kleine. Aus der Kurve ist auch ablesbar, dass ein Widerstand von 0 Ω keinen Nutzen mehr hat, was auch logisch erklärbar ist, da so kein Spannungsteiler gegeben ist (der Zähler der Brüche würde null werden). Zudem sei gesagt, dass der Verlauf nach rechts keine weiteren Extremstellen beinhaltet, weil der Limes der Funktion gegen plus Unendlich gleich Null ist. Damit würde theoretisch jeder noch so große Widerstand die Funktionalität sichern, praktisch ist dies aber durch Störgrößen nicht möglich. Die negativen Funktionsargumente beinhalten auch einen Extrempunkt, für Anwendungen entfällt diese Lösung allerdings, da es weder negative Widerstände noch AD-Bereiche gibt.

Mit einem Shunt von 2,2 kΩ stehen nun also 301 von 1024 Stufen (355 - 656) zur Verfügung. Das bedeutet, dass mehr als ⅔ der Bandbreite nicht genutzt werden. Um das zu verhindern, müssen theoretisch alle Signale unterhalb von 355 subtrahiert und das Ergebnis dann mit Faktor drei multipliziert werden. Diese Signalverarbeitung wird im folgenden Kapitel behandelt.

VI. 4 Der Operationsverstärker

VI. 4. 1 Operations- und Differenzverstärker in der Theorie

Um das Signal wie im vorherigen Kapitel erwähnt zu verändern, werden Operationsverstärker benötigt. Ein OV besitzt neben seinen Versorgungspins zwei hochohmige Eingänge (mit Plus und Minus gekennzeichnet) und einen Ausgang. Er misst kontinuierlich die Spannungsdifferenz der Eingangssignale und gibt sie verstärkt wieder: $U_a = v \cdot (U_+ - U_-)$. Das Signal kann sowohl positiv als auch negativ verstärkt werden. Für die Herleitung der Schaltungen werden theoretische (ideale) OV benutzt, die sich von den praktischen (realen) durch die Kenngrößen unterscheiden. So hat ein idealer Operationsverstärker beispielsweise eine unendlich hohe Differenzverstärkung, ein realer jedoch eine zwischen 10^5 und 10^7. Die reale Offsetspannung liegt je nach Modell zwischen 25 µV und 5 mV, ideal sind natürlich 0 V.

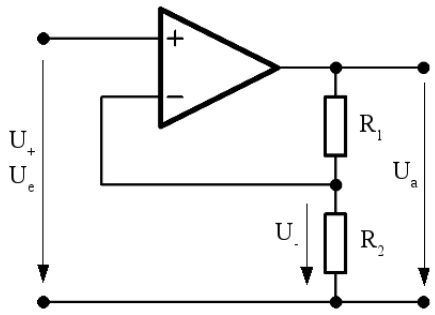


Abb. 1: nicht invertierender Verstärker

Im Grunde können alle nicht idealen Faktoren vernachlässigt werden, solange näherungsweise $U_+ = U_-$ durch die Verstärkung garantiert ist (vgl. [L5], S. 25 ff). Folgend wird nur der nicht invertierende Verstärker betrachtet, wie er in Abbildung 1 dargestellt ist.

Soll also $U_+ = U_- = U_e$ gelten, gilt folglich

$\frac{U_e}{U_a} = \frac{R_2}{R_1 + R_2}$ bzw. $\frac{U_a}{U_e} = 1 + \frac{R_1}{R_2}$ (Spannungsteileregel).

Da der Verstärkungsfaktor *v* das Verhältnis von Ausgangs- zu Eingangsspannung ist, ergibt sich

$v = 1 + \frac{R_1}{R_2}$. Aufgrund der Tatsache, dass U_a ein vielfaches von U_e ist, lässt sich auch schreiben:

$U_a = (1 + \frac{R_1}{R_2}) \cdot U_e$. Der Quotient aus R_1 durch R_2 bestimmt also die Verstärkung, welche allerdings nie kleiner eins sein kann. Diese Schaltung und Formel ist die Basis für den Differenz- oder Instrumentenverstärker. Dabei sind an die Eingängen Spannungsfolger (OV, dessen Ausgang wieder zum negativen Eingang geführt ist) geschaltet. Dadurch wird eine Gleichtaktunterdrückung hervorgerufen und die Eingänge werden hochohmig ausgekoppelt. Den kompletten Schaltungsaufbau zeigt ▷ SL02. Durch die spezielle Verschaltung erfolgt die Berechnung der Ausgangsspannung durch $U_a = (1 + \frac{2 \cdot R_1}{R_2}) \cdot (U_2 - U_1)$ (vgl. [Q3], S. 19). Sie stabilisieren das

Signal und vermeiden sogenanntes Driften. Für diesen Aufbau sind allerdings drei OPV und sieben Widerstände nötig. Dies ist nicht nur eine Platzfrage, sondern auch eine des Aufwandes. Zudem sind die bevorzugten LM 358 doppelter Ausführung in einem 8-poligen PDIP verbaut was zu Folge hat, dass einer nicht genutzt wird.

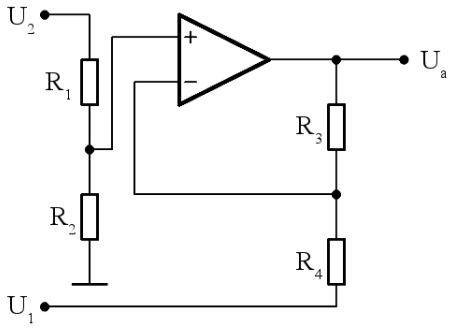


Abb. 2: Subtrahierer

Ein einfacher Differenzverstärker kann auch über *einen* OV als Subtrahierer nachgebildet werden. Ihm fehlt zwar die hochohmige Auskopplung an den Eingängen, was aber durch die Verschaltung des Sensors mit dem Spannungsteiler vernachlässigt werden kann. Ein Subtrahierer besteht aus einem invertierenden und nicht invertierenden Verstärker. Somit berechnet sich die Ausgangsspannung nebenstehender Abbildung 2 nach den Formeln dieser beiden Arten unter Berücksichtigung des Vorzeichens an den Eingängen: $U_a = U_2 \cdot (1+\frac{R_3}{R_4}) \cdot (\frac{R_2}{R_1+R_2}) - U_1 \cdot \frac{R_3}{R_4}$. Für den Fall, dass das Verhältnis der Eingangswiderstände zu den anderen gleich eins ist ($\frac{R_1}{R_4} = \frac{R_2}{R_3} = 1$), ergibt sich

$U_a = U_2 \cdot (1 + \frac{R_3}{R_4}) \cdot (\frac{R_3}{R_4+R_3}) - U_1 \cdot \frac{R_3}{R_4}$ und damit $U_a = (U_2 - U_1) \cdot \frac{R_3}{R_4}$. Das Ergebnis U_a ist also eine um das Widerstandsverhältnis von R_3 zu R_4 verstärkte Differenz der Eingangsspannungen (vgl. [L6] – „Der Subtrahierer“).

VI. 4. 2 Anwendung des Operationsverstärkers

Der gesamten Schaltung stehen mit 4 1,5 V-Batterien theoretisch 6 V zur Verfügung. Da aber die Komponenten nicht ideal sind, sondern auch eine Betriebsspannung verbrauchen, werden folgend 5 V angenommen.

Die Widerstandswerte des Temperatursensors bei -30 °C und +130 °C können dem Datenblatt indirekt entnommen werden. Somit ergeben sich $R_{-30\,°C} = 1230\,\Omega$ und $R_{+130\,°C} = 4114\,\Omega$. Aus der Spannungsteileregel resultieren daher $U_{-30\,°C} = 3{,}21\,V$ sowie $U_{+130\,°C} = 1{,}68\,V$. Folgendes Balkendiagramm (Diagramm 2) zeigt die Auslastung der Spannung für die Temperatur ohne Operationsverstärker.

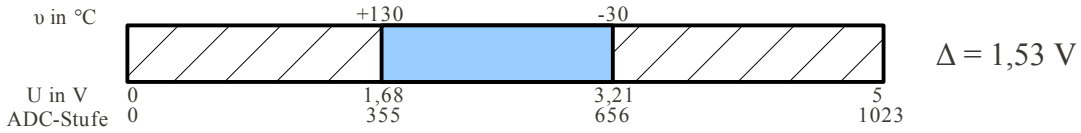


Diagramm 2: gültiger Temperaturbereich bei 5 V ohne OPV

Kommt das im vorherigen Kapitel hergeleitete Prinzip des Operationsverstärkers zur Anwendung, ändern sich die Anfangsspannung des gültigen Bereiches sowie die Ausdehnung und damit der Endwert. Exemplarisch wird 1 V abgezogen und das Ergebnis 2,5-fach verstärkt (Diagramm 3).

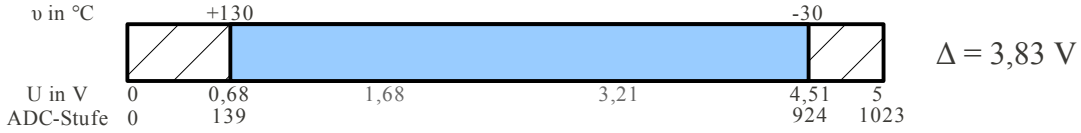


Diagramm 3: gültiger Temperaturbereich bei 5 V mit OPV (Subtraktion: 1 V, Verstärkung: 2,5-fach)

Um die Subtraktion von 1 V zu erzielen, wird diese Spannung an U_1 angelegt. Die Verstärkung bestimmt R_3 zu R_4: $\frac{R_3}{R_4} = 2{,}5$. Die Größenordnung ist theoretisch nicht relevant; es werden für die Anwendung Widerstände im Kiloohm-Bereich genutzt. Das oben maßstabsgetreu gezeichnete Diagramm verdeutlicht, dass auch mit OPV nicht alle Stufen von 0 bis 1023 genutzt werden können. Dies hängt mit elektronischen Störgrößen, wie Offsetspannungen, unerwünschten elektronischen Widerständen usw. zusammen. Jedoch ist der gültige Messbereich um Faktor 2,5 gewachsen, was einen deutlichen Gewinn darstellt.

In der Praxis lassen sich ausgehend von 5 V kaum mit einfachen Mitteln konstante 1 V zur Subtraktion bereitstellen. Daher wird wieder ein Spannungsteiler genutzt. Die Widerstände sind so gewählt, dass bei 5 V ca. 1,5 V Spannungsabfall zwischen R_8 und R_7 besteht (s. ▷ SL03) . Dies hat den Vorteil, dass die Schaltung robust gegen Spannungsschwankungen arbeitet. Beispielsweise differiert die gemessene Temperatur nur um 0,7 °C zwischen 5,5 V und 3,5 V bei 20 °C. Diese dynamische Spannungsquelle garantiert also bei sinkender Ladung der Spannungsquelle (Batterien) nur kleinere Abweichungen.

Der Schaltplan nach bisheriger Konzeption ist unter ▷ SL03 angehängt.

Der zweite im LM 385 integrierte Operationsverstärker kann für den optionalen Drucksensor genutzt werden. Die entsprechenden Widerstände und Komponenten werden auf einer zusätzlichen Platine auflötbar sein.

VI. 5 Temperatur aus dem Widerstand berechnen

Das Datenblatt des Temperaturfühlers KTY-10-6 gibt für die Berechnung der Temperatur die Formel $T = (25 + \frac{\sqrt{\alpha^2 - 4\beta + 4\beta \cdot k_T} - \alpha}{2\beta})\ ^\circ C$ an, wobei $\alpha = 7{,}88 \cdot 10^{-3}$ und $\beta = 1{,}937 \cdot 10^{-5}$. Die Variable k_T beschreibt das Verhältnis des gemessenen Sensorwiderstandes zu R_{KTY} bei 25 °C.
Ein erstes Messprotokoll zur Temperaturberechnung mit dem Sensor ist wie bereits erwähnt in ▷ PR01 dargestellt. Es soll die Funktionsweise erläutern und wurde daher ohne Mikrocontroller erstellt.

Ist die Schaltung ▷ SL03 nach vorhergehendem Kapitel funktionstüchtig, wird an die ClassPad-Schnittstelle ein RS232-TTL-Wandler angeschlossen. Nachdem die UART-Funktion nun vorerst aller zwei Sekunden den digitalisierten Spannungswert ausgibt, soll mit Hilfe der oben angegebenen Gleichung die Temperatur errechnet werden. Sinnvoll ist es, zum Test erst einmal einen Algorithmus in C zu schreiben, der die Berechnungen durchführt. Da die Rechnungen später im ClassPad stattfinden sollen, könnte das Programm mit wenigen Änderungen in C++ überführt bzw. als externe C-Funktion eingebunden werden. Das im Anhang befindliche Programm ▷ QC02 berechnet nach einem eingegebenen ADC-Wert den Widerstand des Sensors R_X, die Variable k_T und die Temperatur T. Wird ein zu kleiner oder zu großer Wert für die Spannung gewählt, gibt das Programm eine Fehlermeldung zurück und zeigt an, welcher Wertebereich Gültigkeit besitzt. Das Programm ist auf eine Schaltung ohne OV konzipiert. Die Grenzen ergeben sich aus dem Inhalt des ADC-Registers bei -30 °C und +130 °C. Da dieses Programm spezielle mathematische Funktionen benutzt, muss neben der Einbindung der *math.h* zusätzlich die Kompilierung mit dem Parameter *-lm* erfolgen. Abbildung 3 veranschaulicht die Arbeitsweise des Programms.

```
user1@AP-20-lin-j:~> gcc -Wall -lm rechnung.c -o rechnung
user1@AP-20-lin-j:~> ./rechnung

ADCW (388 < ADCW < 748) = 200
Wertebereich unterschritten! (min=389)


ADCW (388 < ADCW < 748) = 593
Rx = 2020.539629        kt = 1.010270   T = 26.299127 °C


ADCW (388 < ADCW < 748) =
```

Abb. 3: Kompilierung, Aufruf und Arbeit des Temperaturprogramms

Was in der Theorie gut funktioniert, kann nun auch praktisch realisiert werden. Es gibt nun zwei Fortsetzungsmöglichkeiten: wie oben durch mehrere Formeln die Temperatur berechnen, oder für

jedes Sensorelement eine Funktion erstellen, die in Abhängigkeit eines ADC-Wertes die Temperatur ausgibt. Letzteres Verfahren hat den Vorteil, dass der sensorspezifische Offsetfehler wegfällt und der Taschenrechner nicht viel zu rechnen hätte. Die Vorgehensweise ist recht simpel und wird in ▷ PR02 a) erläutert. Nach dem Protokoll ergibt sich eine Formel, mit der ohne weiteres die Temperatur aus der gemessenen Spannung berechnet werden kann. Als Resultat des Protokolls ist Protokoll 2 b) (▷ PR02 b)) angehängt. Tabelle 1 zeigt Vor- und Nachteile beider Verfahren.

	Datenblattformel	**hergeleitete Formel**
Vorteile	universal für jeden KTY	spezifisch für *eine* Schaltung (Aufbau, Komponenten)
	Schaltungsänderungen haben keinen Einfluss auf die Formel	wenig Rechnung
Nachteile	aufwändige Rechnungen	muss für jede Schaltung erneut hergeleitet werden
	mehrere Formeln nötig	Versuchsanordnungen für Herleitung aufwändig
	kalibriertes Vergleichsthermometer notwendig	

Tabelle 1: Vor- und Nachteile der vorgestellten Temperaturerfassungsverfahren

Zwei Messpunkte für eine Kurvenherleitung zu nutzen ist möglich, aber es gibt dadurch zum Teil große Abweichungen, wie Diagramm 4 (▷ A05) zeigt. Eine einfache Verfeinerung ist die Formelaufstellung über das Polynom $T(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + ... + a_1 \cdot x + a_0$ für n-Messwerte. Die Herleitung einer solchen Formel ist in Protokoll 3 (▷ PR03) dokumentiert.
Ebenso kann dass Temperaturintervall [-30 °C; +130 °C] in n Teile zerlegt werden und jedem Teilbereich wird eine lineare Funktion durch experimentelles Ermitteln zugewiesen.
Allerdings sei gesagt, dass die Verwendung eines Polynoms drei größere Nachteile mit sich bringt. Erstens ist der gültige Messbereich nur so groß, wie die Differenz des größten und kleinsten Wertes zur Herleitung des Polynoms (das zeigt Diagramm 4 deutlich), zweitens ist nur ein kleiner Teil der Funktionswerte annähernd gleich zu den anderen Verfahren (die Graphen für *linear* und *Datenblatt* haben einen ähnlicheren Verlauf) und nicht zuletzt ist die Aufstellung der Formel um einiges aufwändiger.

Um die Transparenz zu erhöhen, findet die Rechnung auf dem ClassPad mit einer selbst herzuleitenden Formel statt. Unterschiede zwischen verschiedenen Temperaturfühlern sind vorhanden.Die Abweichung liegt bei gut 0,5 % (▷ PR04). Auch ist dann die Berechnung der Temperatur mit und ohne OPV gleich. Bei der Datenblattformel müsste zudem k_T angepasst werden.

VI. 6 Auswertung der Temperaturerfassung

Ein wesentliches Problem haben die letzten Kapitel offenbart: die Toleranzen der Komponenten sind in diesen kleinen Messbereichen durchaus signifikant. Hinzu kommen Ungenauigkeiten der Kontrollgeräte und mehrere Möglichkeiten, den Spannungswert in eine Temperatur zu überführen. Auch ist die Berechnung des Temperaturwertes rein im AVR nach der im Datenblatt angegebenen Formel recht aufwändig. Beispielsweise müsste statt der C-Wurzelfunktion *sqrt(x)* aus der *math.h* eine Umgehung mittels Heronverfahren (▷ QC03) nachgebildet werden, da in den gängigen AVR-C-Bibliotheken keine Wurzelfunktion existiert. Die Berechnung ist im ersten Schritt allerdings nicht unbedingt notwendig, da über eine emulierte *atoi()*-Funktion im CASIO-BASIO auch die Überführung von Strings in Integer möglich ist.
Unabhängig von dieser Tatsache hat sich das Verfahren der Temperaturermittlung mit der Herleitung einer schaltungsspezifischen Funktion durchgesetzt. Damit jedoch gewährleistet ist, dass die Module mit jedem KTY-Sensor funktionieren, wird die Eingabe des Funktionsterms am ClassPad erfolgen und nicht fest in den Mikrocontroller einprogrammiert. Für die Übersendung an den ClassPad muss das interne Protokoll für serielle Kommunikation genutzt werden.

Daraus ergeben sich für die Temperaturermittlung folgende Aufgaben für den ATmega:

1. Messintervall, Messzeit und damit Messart vom ClassPad empfangen
2. A/D-Wandlung
3. CASIO-Kommunikationsprotokoll nachbilden
4. den A/D-Wert zwischen 0 und 1023 per UART ausgeben

Damit sind auch nun alle Hardwaregrundlagen abgeschlossen. Fertiggestellt wird der Schaltplan, indem an das UART-Register des Messcontrollers der Sendecontroller mit gleicher Beschaltung der Funkmodule wie in der Belegarbeit angeschlossen wird. Er ist unter ▷ SL04 angehängt. Weiteres dazu wird in den Kapiteln VIII.1, VIII.3 und X.1 erläutert.

VII. Ausgabe auf dem ClassPad

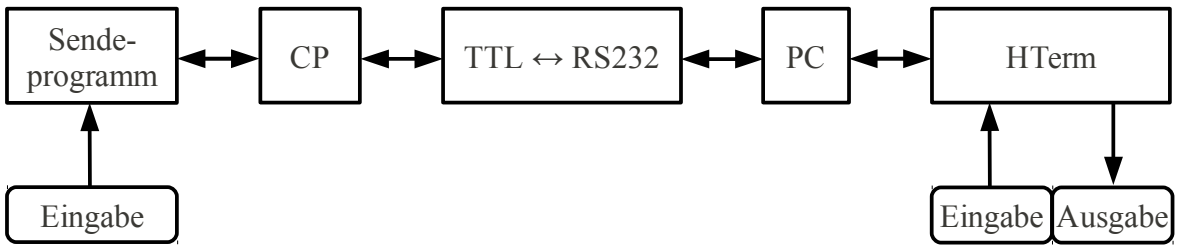
Gemäß der Zielstellung wird nun in diesem Kapitel die Ausgabe der gemessenen Daten auf dem ClassPad erfolgen. Der erste Abschnitt beschreibt das CASIO-Kommunikationsprotokoll und dessen Abbildung auf einem AVR, danach wird die Darstellung der Temperatur in CASIO-BASIC erfolgen.

VII. 1 ClassPad Kommunikationsprotokoll

In der Belegarbeit spielte es nur eine untergeordnete Rolle, da der umfangreiche Teil des Protokolls entfallen konnte. Um vom ClassPad her Aktionen an einem Mikrocontroller auszulösen, musste dieser nur auf bestimmte Zeichen reagieren. Dabei war die Interpretation von Header-Files, Prüfsummen usw. nicht weiter von Bedeutung. Nun folgt der umgekehrte Weg: der AVR beginnt die Kommunikation und muss dem korrespondierenden Taschenrechner entsprechende Informationen generieren.

VII. 1. 1 Analyse der Kommunikation

Mit der in der Belegarbeit genutzten Abhörschaltung wurde der Datenverkehr bei Kommunikation sichtbar und der Inhalt daraufhin analysiert. Protokoll 5 (▷PR 5) stellt dazu mehrere Beispiele dar, von denen eines exemplarisch betrachtet wird. Den Aufbau skizziert Zeichnung 1:



Zeichnung 1: Kommunikationsschema von ClassPad zum PC

Folgend soll über ein kleines CASIO-BASIC Programm (▷ QC04) der String *856* übertragen werden. Das Signal wird dann auf RS232-Definition gebracht und vom Computer über HTerm ausgegeben. Dort erfolgt auch die Interaktion, um die Daten zu empfangen.

Die abgebildete Darstellung der abgehörten Daten ist hexadezimal.

```
CP:   15          } Handshake
PC:   13          }
CP:   3A 4E 44 64 00 01 00 01 00 06 00 06 05 FF F8 } NDd-Header und Bestätigung
PC:   06          }
CP:   3A 00 01 38 35 36 00 5C } Datenstring und Bestätigung
PC:   06          }
CP:   00  - Ende
```

Der NDd-Header enthält u.a. Informationen zur Datenlänge und -art (z.B. String, Programm, Zahl). Im Datenstring stehen grün markiert die drei Ziffern in der hexadezimalen ASCII-Entsprechung, die Prüfsumme ist blau gefärbt.

$38_h = 56_d \triangleq 8_{ASCII}$, $35_h = 53_d \triangleq 5_{ASCII}$, $36_h = 54_d \triangleq 6_{ASCII}$

VII. 1. 2 Theorie der CASIO-Prüfsumme

Nach Recherchen setzt sich die Checksumme wie folgt zusammen:

> „Alle Zeichen im Datenstring (auch die Prüfsumme selbst) werden ihren hexadezimalen Werten nach addiert und durch 256 dividiert. Das Modulo muss dann $3A_h$ ergeben.“ (vgl. [L8], 1. Beitrag)

Daraus ließe sich durch Umstellen die Prüfsumme errechnen, jedoch gibt es einen einfacheren Weg: von $3A_h$ wird die Summe aller Hex-Werte (ohne Prüfsumme) abgezogen (vgl. [L8], 2. Beitrag) und anschließend wird $FFFFFF00_h$ subtrahiert, um den Überlauf des 8 Bit Registers – hervorgerufen durch das negative Ergebnis – rückgängig zu machen.

Die Prüfsumme muss für jeden Kommunikationsvorgang neu errechnet werden und soll Fehler im Datenaustausch anzeigen. Je nach Komplexität kann auch eine (einfache) Fehlerkorrektur durchgeführt werden. Die simpelste Variante ist ein XOR aller Bits. Kippt allerdings eine gerade Anzahl an Bits während der Kommunikation, ist aus der Prüfsumme keine Fehlübertragung lesbar.

VII. 1. 3 Erzeugen einer Prüfsumme

Um das hexadezimale Zahlenpaar zu errechnen, wird nach dem zweiten Beispiel der vorherigen Ausführung vorgegangen. Der zugehörige Quellcode (▹ QC05 Z. 27 - 188, ▹ PAP02 a); für Prüfsummenberechnung s. ▹ QC05 Z. 52 - 87) sei hier kurz erläutert:

Nach Anlegen der benötigten Variablen in ihren Datentypen wird der vom A/D-Wandler kommende Spannungswert durch Ganzzahldivision in seine Ziffern zerlegt. Jede Ziffer wird dabei einer Variablen zugewiesen. Anschließend erfolgt die Addition von 48. Dies ist der ASCII-Wert der Null. Damit wird die Zahl auf den dezimalen ASCII-Wert gebracht, um später die Prüfsumme zu errechnen. Zuvor werden alle Nullen vor dem Ergebnis mit *0x00* ersetzt. Dies ist nötig, da sonst die Prüfsumme falsch berechnet wird. Nachfolgend wird nach o.g. Prinzip die Prüfsumme erstellt.

Nun beginnt die eigentliche Kommunikation auf umgekehrten Wege wie in VII.1.1 dargestellt:

Über die UART-Funktion wird eine 15_h ausgesandt. Antwortet ein ClassPad mit 13_h, wird der NDd-Header Stück für Stück gesendet. Einer Bestätigung durch die Gegenstelle folgt der Versand des Datenstings, der immer mit $3A_h$ 00_h 01_h beginnt. Ist die Zahl einstellig, folgt die Ziffer und dann drei leere (0x00) Felder. Bei zwei Ziffern entsprechend die Ziffern und zwei Nullzeichen. Nach gleichem Prinzip erfolgt die Kommunikation bei drei Ziffern. Besteht der ADC-Wert jedoch aus vier Zeichen, dann benutzt das Protokoll einen anderen NDd-Header und acht Datenfelder (davon sieben nutzbar). Abschließend wird die Prüfsumme angehängt und nach Bestätigung des Dateneinganges die Verbindung beendet.

Die Abbildung 4 zeigt, dass sich nun die Ausgaben in HTerm von ClassPad und PC gleichen:

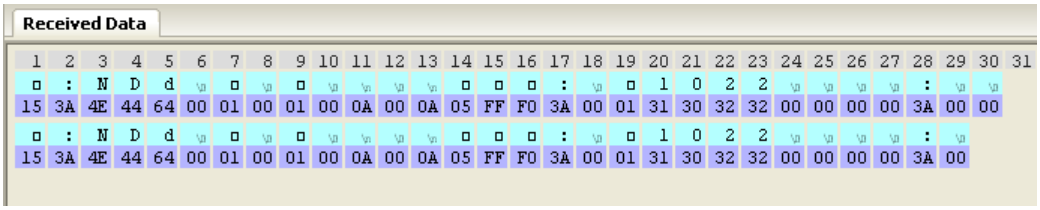


Abb. 4: Empfang des CASIO-Kommunikationsprotokolls; oben: CP → PC, unten: µC → PC

Das in der Abbildung 4 dargestellte Nullzeichen der oberen Zeile an Position 30 ist nicht in Verbindung mit dem Kommunikationsvorgang entstanden, sondern weil das Programm noch beim Umstecken der Drähte aktiv war und bei Einstecken von Tristate-führenden Drähten nicht definierte Zeichen übertragen werden können.

Anmerkung zur Programmierung:

Die Funktion *uart_puts();* greift intern zyklisch auf *uart_putc();* zu, um den Stringinhalt einzeln zu senden. Daher ist es nicht von Bedeutung welche Funktion genutzt wird. Jedoch ist *uart_putc();* einfacher zu lesen, zumal der String als solches nicht aus ASCII-Zeichen besteht und daher vorher hexadezimal definiert werden müsste.

VII. 1. 4 Datenempfang vom ClassPad

Vor dem Start einer Messung müssen die Parameter festgelegt werden. Dabei handelt es sich um Messintervall und -zeit, die vom Benutzer auf dem ClassPad eingestellt werden können (bzw. bei der Sofortmessung fest sind). Dazu dient die Funktion *cp_get()*. Sie empfängt den NDd-Header und verwirft ihn wieder. Danach werden die Daten eingelesen und in je eine Variablen gespeichert. Ebenso wird mit der Prüfsumme verfahren. Die Daten werden zur jeweiligen Zahl zusammengesetzt und dann an das Hauptprogramm übergeben.

Die Funktion *cp_get()* ist als Bestandteil in der Subroutine *casio.c* enthalten (▷ QC05, ▷ PAP02). Verwendet wird sie im Hauptprogramm des Messcontrollers ▷ QC06.

VII. 2 Ausgabe im ClassPad-Programmiermenü

Im letzten Schritt erfolgt nun die Ausgabe der vom Mikrocontroller gesendeten Daten auf dem ClassPad. Dies geschieht in der im Taschenrechner integrierten Programmierumgebung, in der auch die serielle Kommunikation möglich ist.

Elementar ist die Frage, ob die Anzeige in CASIO-BASIC oder als C++-Add-In erfolgen soll. Die Wahl fiel hier auf die integrierte Programmumgebung. Grund dafür ist eine einfache Weitergabe zwischen den Taschenrechnern, was bei einem Add-In nicht der Fall ist, da es extra von einem Computer einzeln auf die Taschenrechner installiert werden muss. Ebenfalls entfallen auch mögliche Zusatzprogramme, da das komplette Anzeige-Programm im CASIO-BASIC Befehlssatz geschrieben wird. Ob der Umfang der Sprache auf allen OS-Versionen gleich ist, wäre anhand der verschiedenen Modelle einzeln zu prüfen. Aktuell wird mit jedem Neugerät mindestens die Version 3.03 (s. [L9] „Feature Übersicht“) verkauft. Die Version 2.5.x, die zu Beginn des Schuljahres 2007/2008 Standard war, ist nur noch auf wenigen Taschenrechnern vorhanden, wie Diagramm 5 (▷ A06) zeigt.

Das Programm soll über folgende Funktionen verfügen (in der Bedienungsanleitung erklärt):

- eine Sofortmessung
- Diagrammmessung
- Einstellungen für Berechnungspolynom und Anzeige sowie Aktualisierungsrate
- Löschen aller Variablen; Löschen aller Einstellungen und Variablen
- Verwendung von eingegebenen Standardwerten für die Anzeige (nur Diagramm)

VII. 3 ClassPad Programmierung – Probleme und Lösungen

In der Programmierung gibt es meist mehr als nur einen Weg, um zum Ziel zu kommen. Es gilt hier ein möglichst ausgefeiltes Programm zu entwickeln, das nicht nur sicher arbeitet, sondern auch wenig Speicherplatz einnimmt. Die Funktionsweise des Programms ist in ▹ PAP03 grafisch dargestellt. Um das Programm so umzusetzen sind einige Umwege nötig, die hier im folgenden kurz beschrieben sind. Vorweg ist noch anzumerken, dass die Befehlsreferenz des Taschenrechners im Handbuch zwar 417 Kommandos umfasst, jedoch davon nur wenige genau beschrieben sind.

Um die Übersicht zu gewährleisten und Redundanz durch den Einsatz gleicher Funktionen in verschiedenen Programmteilen zu eliminieren, nutzte das Programm anfangs 11 Unterprogramme, nach den Analysen nunmehr 18. Zu komplexeren Programmen ist je ein Programmablaufplan unter ▹ PAP04/*Name* angehängt. Die entsprechenden Quellcodes sind unter ▹ QC07/Name verzeichnet.

Die Auswahl der gewünschten Aktion im Hauptmenü erfolgt über die Eingabe einer zugeordneten Zahl. Diese Erfassung der Eingabe ist einfacher auswertbar als ein Vergleich der xy-Koordinaten einer Stifteingabe, der auch ein gewisser Streuradius zugewiesen werden müsste (▹ QC07/temp1).

Das erste Problem ergab sich aus der Anzeige auf dem ClassPad: die per UART empfangenen Zeichen werden aufgrund des NDd-Headers mit dem Datentyp STR (String) abgelegt. Jedoch kann mit einem String keine Rechnung vollzogen werden. Es gibt zwar die Möglichkeit, einzelne Zeichen – insofern es Zahlen sind – in eine Zahl zu überführen, für Zahlenketten gibt es allerdings keinen Befehl. Somit muss die aus C bekannte *atoi()*-Funktion nachgebildet werden (▹ QC07/atoi).

Die vorgesehene Möglichkeit einen Standard zu definieren und zu verwenden führt dazu, dass das Programm sehr umfangreich wird. Jedoch lässt die Variable, in der die Verwendung des Standards abgelegt wird, weitere Optionen zu. An ihrem Vorhandensein kann z.B. ein erster Programmstart festgestellt werden und durch entsprechende Hinweise dem Nutzer eine kleine Hilfestellung zur weiteren Verfahrensweise gegeben werden.

Werden statt If-Then-Else-Anweisungen bei Mehrfachauswahlen die geeigneten Switch-Case-Strukturen benutzt, kommt es in einigen Fällen zu nicht gewünschten Sprüngen in andere (Unter-)

Programmteile. Um dennoch die übersichtliche Programmierung zu gewährleisten, wurden kleine Zwischenprogramme geschrieben, die eine Statusvariable der vorherigen Instanz auswerten und die entsprechenden Unterprogramme der Reihe nach aufrufen (▷ QC07/diagr, /polyprob, /wert)

Ist ein Setzen der Statusvariablen nicht möglich weil im Gegenzug das Programm über die Maßen komplex würde, wird auf die If-Then-Else-Anweisung zurückgegriffen.

Um benutzerdefinierte Informationen für die Messung zum AVR zu übertragen, wird ebenso das integrierte Kommunikationsprotokoll genutzt. Wird ein Zeichen gesendet, so spielt der Datentyp eine wesentliche Rolle. In der Belegarbeit wurde dieser mit *InputStr x* als String vorgegeben. Bei *Input x* wird das Format an den eingegebenen Werten festgemacht. Bei Zahlen ist es beispielsweise der Typ *NUM*. Sendet man ein solches Zeichen, so wird ein anderer NDd-Header als bei Strings genutzt. Zudem ist die Formatierung anders als in Kapitel VII.1.1 dargestellt. Da aufgrund der Erfahrung und des bestehenden Quellcodes weiterhin mit Variablen vom Typ *STR* und nicht *NUM* gearbeitet werden soll, müssen diese mithilfe von *ExpToStr num_var, string_var* explizit gecastet werden. Die Eingabe muss weiter mit *Input x* erfolgen, da eine Zeichenkette nicht direkt mit Zahlen verglichen werden kann. Dies wiederum ist aber nötig, da das ClassPad-Programm Abfragen durchführt, um Fehleingaben zu erkennen und entsprechend zu reagieren. Die Typenumwandlung wird daher nach den Vergleichen am Programmende durchgeführt. Fehleingaben sind beispielsweise nicht zulässige Werte für die Temperaturdarstellungsgrenzen im Diagramm oder bei der Messzeit. (▷ QC07/e_interv, /e_zeit, /e_obere, /e_untere)

Die Variablen für Messintervall und -zeit können als kleinsten Wert *2 s* annehmen. Ursache ist die Verarbeitungszeit des BASIC-Programms. Der Mikrocontroller könnte ohne weiteres auch in kürzeren Zeitabständen eine Wandlung durchführen. Hinzu kommt aber, dass die Funkübertragung eine gewisse Latenzzeit besitzt. Im Extremfall sendet der Mikrocontroller den Handshakestart der nächsten Zahl, der ClassPad ist aber noch nicht zum Variablenempfang bereit, da die Ausgabe der letzten noch in Bearbeitung ist. Zur Synchronisation wären kleine Pakete notwendig. Die kleinste Einheit die vom ClassPad gesendet werden kann ist jedoch ein String mit drei Zeichen. Diesen wiederum auszuwerten und dann das Ergebnis der A/D-Wandlung zu senden würde länger dauern, als im geeigneten Moment gleich den ADC-Wert zu übertragen.

Um diesen Moment abzuwarten, wartet der Mikrocontroller nach jeder Datenausgabe $Messintervall \cdot 1000\,ms$ ab (s. ▷ QC06 Z.108 ff.). Ein Abzug der 100 ms aus *cp_get()* muss nicht beachtet werden. Eingeführt wurde die Wartezeit, damit bei der Parameterabfrage nicht das o.g.

Fehlverhalten eintritt und solche Sicherheitszeiten nicht bei jedem Funktionsaufruf vom Programmierer einbezogen werden müssen (s. ▷ QC05 Z. 274).

Als schwerwiegendstes Problem hat sich die Beendigung der Funktion „Diagrammmessung“ herausgestellt. Die eigentliche Komplikation ist, dass die Rückkehr zur Hauptauswahl nicht einfach umzusetzen ist, weil eine Auswertung der erfassten Messwerte möglich sein soll. Wird das Programm über die Icon-Leiste unten beendet, bleiben noch Variablen gespeichert, die bei späteren Rechnungen im Main-Menü (auch unabhängig vom Programm) die Ergebnisse verfälschen. Also müsste das Programm abgebrochen und wieder geöffnet werden, um mit einer Auswahl die entsprechenden Variablen zu löschen. Dies ist allerdings kein benutzerfreundlicher Weg und daher auch nicht anzustreben.

Eine Möglichkeit ist es mittels sogenanntem Polling in einer Schleife abzufragen, ob eine Taste gedrückt wurde. Die programmiertechnische Umsetzung ist sehr einfach (▷ QC08). Allerdings hat der Lösungsansatz zwei gravierende Folgen:

- der Prozessor ist nur noch damit beschäftigt, das Polling durchzuführen; bis die Taste nicht gedrückt wurde ist keine Aktion ausführbar (auch kein Scrollen in den angezeigten Werten)
- aufgrund der Dauerauslastung des Prozessors steigt der Stromverbrauch an.

Gerade wegen des ersten Punktes kommt der Lösungsansatz keinesfalls in Frage.

Die einzig annehmbare Lösung zum Polling oder anderen Möglichkeiten ist das Interrupt. Das heißt, dass das Programm nicht darauf wartet, das seine Aktion ausgeführt wird, sondern alle Ereignisse registriert und bei einem bestimmten eine bestimmte „Unterbrechung“ ausgeführt wird. Im Gegensatz zum Polling fallen die o.g. Punkte weg.

Da die Programmiermöglichkeiten hier aber einen Riegel vorschieben, muss die Aktivierung des Interrupts etwas abgewandelt werden.

Ein Interrupt wird beim ClassPad allgemein ausgelöst durch:

- Tastendruck,
- Klick auf den Bildschirm:
    - an definierter Stelle oder
    - Button in einem Fenster.

Die erste Variante scheidet aus, da sie wegen der begrenzten Programmiermöglichkeiten in Hinblick auf die Auswertung wieder auf Polling hinausläuft.

Der erste Unterpunkt der zweiten Variante ist keine Lösung, da die Abfrage ebenfalls ein Polling ist: Es müsste ein Bereich angezeigt werden, auf den geklickt werden soll und ständig eine Abfrage durchgeführt werden, ob der Bereich angeklickt wurde (= Polling). Die beschriebene Methode ist grundsätzlich im CASIO-BASIC möglich, aber wegen ihres Aufwandes und Nachteils keine Lösungsmöglichkeit.

Bliebe noch die Variante, ein Fenster mittels *Message* einzublenden. Wird die Frage (z.B. „Zurück?“) mit OK bestätigt, läuft das Programm weiter (in dem Falle springt es zur Hauptauswahl zurück). Neben dem Nachteil, dass das Fenster immer hin und her geschoben werden müsste, da es entweder das Diagramm oder die Ausgabe verdeckt, verhält es sich ähnlich wie beim Polling: solange es nicht aktiviert wurde, ist ein Scrollen in den Werten nicht möglich.

Eine weitere Hürde ergibt sich bei der Anzeige der Werte, die mit der Rückkehr ins Hauptfenster im Zusammenhang steht.

In der oberen Hälfte wird das Diagramm gezeichnet, während in der unteren im eingestellten Intervall die Messwerte angezeigt werden. Grundsätzlich kann die Ausgabe der Werte auf zwei Wegen erfolgen:

1. die Werte untereinander ausgeben (per *Print x*)
2. in einer Art Tabelle mit der zugehörigen Zeit (mit *Local* die Positionen explizit angeben)

Der erste Weg ist weniger benutzerfreundlich, denn der Befehl *Print* kann nur ein Argument verarbeiten, so beispielsweise die Ausgabe einer Zahl *oder* einer Variablen, Kombinationen sind in einem Befehl nicht möglich. Somit kann die Zeit nicht angezeigt werden, zu der der Messwert ermittelt wurde.

Bei der zweiten Variante gibt es noch ein größeren Nachteil: das ClassPad-Display hat 160 vertikale Pixel. Steigt der y-Wert von *Local* über 160 bricht das Programm mit einem Fehler ab. Zwar könnte ab „160“ von vorn angefangen werden die Daten anzuzeigen, jedoch werden dadurch ältere überschrieben.

Letztendlich wird nun so verfahren, dass nach einer Messung ein Fenster erscheint, das zum Beenden des Programms und Öffnen des Statistik-Menüs auffordert, wo dann die Zeitargumente in *list1* und die entsprechenden Temperaturen in *list2* automatisch eingetragen werden (vgl. ▹ QC09). Über *Calc* können dann z.B. Regressionen und weitere Auswertungen der Datensätze erfolgen. Damit wird auch eine von Anfang an präferierte Verfahrensweise genutzt.

Zuerst war für das Bilden der Listen der Befehl *augment* verwendet worden. Allerdings hat sich herausgestellt, dass das Programm – je länger die Liste wird – länger braucht, um wieder den ankommenden ADC-Wert abzufragen. Da allerdings die Liste *Zeit* mit dem *seq*-Befehl erstellt werden kann, und nur noch die Liste der Temperaturen mit augment erzeugt wird, ist es relativ unwahrscheinlich, dass sich bei kleinen Messzeiten und Messintervallen ClassPad und Messerfassungseinheit verfehlen. Ein Test ergab, dass bei zwei Sekunden Messintervall noch 72 Sekunden (37 Messungen) lang gemessen werden kann. Bei einem Intervall von drei Sekunden sind es schon 198 Sekunden (67 Messungen).

VII. 4 Zusammenfassung der ClassPad-Programmierung

Nach diesem Kapitel ist nun ein Programm entstanden, das den gewünschten Ansprüchen entspricht und durch die transparent in Unterprogrammen durchgeführte Struktur leicht erweitert oder umgeschrieben werden kann, um es auf andere Messungen abzustimmen. Viel Erfahrung in der ClassPad-Programmierung macht es möglich, alle Wege auszuprobieren und den besten zu wählen. Zum Teil stellte sich aber später heraus, dass doch eine andere Variante genommen werden muss, weil ein Weiterkommen sonst unmöglich erscheint (z.B. Festlegen des Variablentyps: NUM/STR, Beenden der Diagrammmessung).

Grundlegend ist der Befehlssatz für die Ausgabe der gewünschten Werte nicht sehr flexibel, wie beispielsweise *printf()* in C. Die Ausgaben sind vorformatiert und es muss immer bei einem neuen Typ (Variable, String,...) ein neuer Ausgabebefehl genutzt werden.

Jedoch konnten alle benötigten Funktionen in CASIO-BASIC programmiert werden, ohne dass eine Einarbeitung in C++ notwendig war, um ein Add-In zu schreiben.

VIII. Integration zum Produkt

Alle theoretischen Grundlagen und zum Teil auch praktische Erfahrungen liegen nun vor, um die Platine aus den einzelnen Komponenten zu einem Gerät zusammenzufügen.

VIII. 1 Verteilung der Aufgaben

Die Aufgaben sind im Blockschaltbild ▹ A07 dargestellt. Sie orientieren sich an einer klaren Trennung von Kommunikationsprozess (Funk), Messparameter- bzw. Datenerfassung und Datenauswertung sowie -anzeige. Die Steuerung der Funkmodule und die Temperaturerfassung wurden bereits in Hardware auf zwei Mikrocontroller aufgeteilt, damit zum einen eine klare Strukturierung erreicht wird, zum anderen reichen die vorhandenen Portbits des ATmega168 nicht aus, um genug Diagnosemöglichkeiten zu bieten. Einige Pins wären zudem mehrfach besetzt (z.B. liegen die LEDs des Funkstatus im Register C (ADC2 bis ADC5) an. Da jedoch der Funkquellcode als auch die A/D-Wandlung auf die ATmgega44/88/168-Reihe bereits optimiert ist, wird dieser Controller weiterhin verwendet.

VIII. 2 Steuercodes

Um dem Messcontroller mitzuteilen welche Art der Messung durchgeführt werden soll, wurde die Variable *messtyp* eingeführt (vgl. ▹ QC5, Z. 66 - 69). Sie wird mit '0' initialisiert und erhält nach der Abfrage der Messparameter als Zuweisung entweder Null (Sofortmessung) oder Eins (Diagrammmessung).

Theoretisch kann jedes Zeichen diese Änderung bewirken, allerdings ist die Funktion *cp_get()* aufgrund der üblichen Anforderungen so konzipiert, dass sie nur Zahlen von 0 bis 999 auswerten kann. Da wie im Kapitel VII. 3 beschrieben '2' der kleinste Messabstand ist, wurde für die Übertragung des Messintervalls bei einer Sofortmessung die Variable *tdif = 0* gewählt. Der Messcontroller erkennt dieses Zeichen als Steuerzeichen, da es in der Diagrammabfrage nicht als Parameter vorkommen kann. Für eine ähnliche Verwendung ist *tdif = 1* reserviert. Zudem eröffnet diese Vorgehensweise 2000 Steuercodes: wird *tdif* als Steuerflag erkannt (0 oder 1), so kann im Ausdruck *zeit* eine Zahl von 0 bis 999 weitere Anweisungen auslösen. Das ist auch der Grund,

warum erst *tdif* und dann *zeit* abgefragt werden (s. QC5, Z. 63 f.).

VIII. 3 Schaltplan, Platinenlayout und Gehäuse

Der Schaltplan ist unter ▹ SL04 angehängt. Er ist nach dem in Kapitel VIII. 1 vorgestellten Blockschaltbild konzipiert. Zudem erfolgt eine Trennung zwischen analoger und digitaler Masse (vgl. [L11], S 96). Nach dem ersten Entwurf sollten alle Diagnoseanzeigen (LEDs) zusätzlich angeschlossen werden können und nicht Bestandteil der Platine sein, um Strom zu sparen. Jedoch hat sich nach einem Test herausgestellt, dass die LEDs des Funkmoduls immer mit diesem verbunden sein müssen, da sonst keine Kommunikation über Funk möglich ist.

In einer weiteren Überprüfung hat sicher herausgestellt, dass die angedachten 6 V Betriebsspannung zu hoch sind und bei Diagrammmessungen für Controllerresets sorgen. Daher wurde eine Batterie entfernt und überbrückt. Der Fehler trat bei Test zu Hause nicht auf, da dort (um die Akkus/Batterien zu schonen) mit einem Labornetzgerät mit 4 bis 5 V gearbeitet wurde.

Der komplette Funk-Komplex ist abschaltbar. Das spart zum einen Strom zum anderen ist eine serielle Verbindung direkt vom Messcontroller zum Taschenrechner nur möglich, wenn es keinen anderen Empfänger gibt. Umgekehrt gilt das gleiche: der ClassPad kann nicht direkt angeschlossen werden und ein weiterer über Funk Daten erhalten.

Für den Anschluss der Experimentierplatine wird eine 15-polige Sub-D-Buchse verwendet, weil 14 verschiedene Signale übertragen werden müssen. Ein Pin bleibt damit unbesetzt.

Da der Schaltplan mit EAGLE erstellt wurde, war es ohne weiteres möglich, aus der Schaltung die Platine zu layouten. Das Programm hat dies zu 94 % geschafft, die restlichen 6 % wurden manuell geroutet. Aufgrund zunehmender Komplexität der Schaltung mit geringem Spielraum mussten zwei Drahtbrücken eingebaut werden.

Das Layout wurde dann mittels UV-Belichtungsverfahren auf eine fotopositive Epoxyd-Platine übertragen, mit Natriumhydroxidlösung entwickelt und mit Eisen-III-Chlorid-Lösung (aq) geätzt.

Das Layout ist in Originalgröße unter ▹ S02 angehängt.

Da das Layout in der endlichen Form mit den Abmessungen 78x82mm recht ungewöhnlich ist, dazu noch ein Batteriefach benötigt, ist der Selbstbau eines Gehäuses der einzig akzeptable Weg. Nach

einer groben Skizze wurde die Zeichnung mittels QCAD und Houdini technisch umgesetzt (▷ S03) und im Anschluss aus Hartpapierplatten gefertigt.

VIII. 4 Rechtliche Einordnung und Lizenz

Die Quellcodes für die Ansteuerung der UART-Schnittstelle stammen von Peter Fleury und sind in der vorliegenden Version vom 1. Juli 2007 (uart.h v1.8.2.1, uart.c v1.6.2.1) unter der GNU General Public License veröffentlicht. Diese erlaubt die freie Verbreitung und Modifikation des Quelltextes. Er muss allerdings lesbar weitergegeben werden (kein Maschinencode) (vgl. [L13], § 5).

Da das Programm für die RFM-12 Sende- und Empfangsroutine von Benedikt K. ebenfalls auf dem Quelltext von P. Fleury basiert, gelten die gleichen Rechte für Verbreitung und Abänderung. Zudem ist vom Autor eine kommerzielle Nutzung untersagt (vgl. [K3], 20.10.2009), die die GNU GPL nicht ausschließt.

Um die Rechte und Interessen der Autoren zu sichern und zu wahren, läuft dieses Projekt unter GNU GPL mit der Einschränkung der nicht kommerziellen oder wirtschaftlichen Nutzung. Damit ist eine freie und unentgeltliche Verbreitung sowie Modifikation garantiert.

Es gilt die GNU GPL der Version 3 in Ausfertigung vom 29. Juni 2007. Ein entsprechender Hinweis auf die Quelle des kompletten Textes als auch die rechtliche Geltendmachung durch die Passage im Quelltext sind eingefügt. Zudem liegt im Verzeichnis unter *gpl-3.0.txt* eine rechtsgültige Kopie der Lizenz von gnu.org in englischer Sprache vor.

IX. Evaluierung

IX. 1 Kriterien aufstellen

Um die nun fertig entwickelten Komponenten zu testen, wurden Evaluierungen durch die Chemielehrerin und Mitschüler durchgeführt. Bei den Tests mit Mitschülern musste aufgrund fehlender Chemikalien mit Alternativen gearbeitet werden, die im Endeffekt aber die gleiche Wirkung erzielten. Die Protokolle der Tests sind unter EV01 ff. angehängt.

In der Akzeptanzanalyse wird mit folgenden Eigenschaften geprüft, ob das Produkt den

Anforderungen an den Chemieunterricht genügt:

Unter dem Punkt *Hardware* werden Äußerlichkeiten der Komponenten erfragt, die für den Einsatz nicht unerheblich sind. Ebenso ist es wichtig ob die Bedienung auch ohne Handbuch im Groben möglich wäre (*Selbsterklärend*), ob die angezeigten Informationen eher störend als sinnvoll sind, ob die Anordnung der Anschlüsse logisch ist und die Dimension auch für den Unterricht geeignet sind. Zudem ist die Aufbauzeit ein wichtiger Faktor, denn an sich lässt sich ein einfaches Thermometer schneller einschalten und benutzen.

Die Kategorie *Software* zielt auf die Bedienung des ClassPad-Programms ab. Hier kann festgestellt werden, ob die Ein- und Ausgaben verständlich sind, ob das Programm an sich eine für den Nutzer schnell zu verstehende Struktur besitzt, wie zuverlässig das Programm arbeitet und ob es den angedachten Zweck bei den Auswerteoptionen letztlich erfüllt. Auch wichtig ist die Einarbeitungszeit, da die Funktionsweise im Vergleich zu einem normalen (digitalen) Thermometer anders ist, dafür aber mehr Möglichkeiten bietet.

Der Punkt der Störfälligkeit gehört zu beiden Kategorien, äußert sich jedoch erst in der Software. Dabei geht es darum, ob der Sensor in verschiedenen Substanzen gleich gut arbeitet.

Der dritte Aspekt behandelt das Handbuch, das dem Anwender letztendlich als einzige Referenz dienen kann. Essentiell ist eine verständliche, nicht komplizierte aber dennoch kurze Anleitung, damit der Nutzer den Überblick behalten kann.

Die Bewertung erfolgt nach verbalen Vorgaben und zusätzlich in Notenpunkten, damit das Kriterium besser eingeschätzt werden kann.

Die Akzeptanzanalysen und jeweils eine individuelle Auswertung sind unter EAA bzw. EA verzeichnet.

IX. 2 Auswertung der Tests

Dieses Kapitel wertet kurz alle Analysen aus. Spezifische Untersuchungen zu den Anwendertests sind in jeweiligen Protokollen zu den Evaluierungen beigefügt. Da von Test zu Test aufgrund der Resultate Weiterentwicklungen an den beanstandeten Teilen vorgenommen wurden, sind die Experimente nicht direkt vergleichbar. Dieser Weg ist allerdings der günstigste, um nicht verschiedene Bewertungen einzuholen, Änderungen vorzunehmen und wieder Anwendertests durchzuführen.

Im Allgemeinen sind die Bewertungen positiv, lassen aber auch Spielraum für Verbesserungen, die alle im Softwareteil durchgeführt wurden.

Jeder Test hat auf seine Weise zur Optimierung des Programms beigetragen. Damit wurde zwar die Komplexität und Größe um ein vielfaches höher als in der Ursprungsversion, jedoch wächst gleichermaßen der Grad an Selbsterklärung und Verständnisprobleme bei der Programmnutzung nehmen ab.

Zusätzlich wurde nach der vierten Analyse im Zuge der Programmsicherheit das Unterprogramm *polyprob* erstellt, welches prüft, ob ein zulässiges Polynom eingegeben wurde und entsprechend reagiert.

Zudem wurde in ▹ PR06 a) und b) der Vergleich zwischen dem entwickelten Messsystem und dem bestehenden von CASIO (EA-200) durchgeführt und ausgewertet. Er hat gezeigt, dass das hier konzipierte Gerät durchaus mit dem kommerziellen Produkt von CASIO mithalten kann und auch recht genau die Werte ermittelt.

Die durchgeführten Akzeptanzanalysen haben auch immer gezeigt, dass ein Programm nie allen Anwendern gerecht werden kann und damit im Prinzip nie fertiggestellt werden kann.

X. Ausblick und Fazit

X. 1 Experimentierplatine

Um die Modularisierung gemäß der Aufgabenstellung zu gewährleisten, ist der Anschluss einer Experimentierplatine möglich. Dabei verlaufen drei Pins der Platine direkt zu den A/D-Anschlüssen, der vierte wird zu dem zweiten Operationsverstärker des LM 358 geleitet und von dort aus zum Mikrocontroller geführt.

So kann beispielsweise ein Drucksensor angeschlossen werden, in dessen Kombination mit dem Temperaturfühler das „Prinzip des kleinsten Zwangs“ nach Le Chatelier nachweisbar ist. Die Integration eines Drucksensors ist im Anhang (▷ A08) in der Theorie erläutert.

Ein weiteres Beispiel zur Modifikation bietet das in Jahrgangsstufe 13 behandelte Feld der Elektrochemie. Dabei werden oft Spannungen gemessen, was über einen Spannungsteiler auf der Platine recht einfach ist.

Vom Messcontroller führen zudem weitere vier Anschlüsse des Ports B (PB0 – PB3), die mit Leuchtdioden oder Tasten besetzt werden können, zur Platine. Je nach Datenrichtung sind Diagnosen über binäre Anzeiger oder boolesche Anzeigen (z.B. von über-/unterschrittenen Temperaturgrenzen) möglich.

X. 2 Verbesserungsvorschläge und weitere Möglichkeiten

Während der Arbeit haben sich viele Möglichkeiten der Verbesserung ergeben. Einige konnten durchgeführt werden, andere würden den angelegten Rahmen sprengen bzw. passen nicht zur Aufgabenstellung. Trotz dessen seien an dieser Stelle einige aufgeführt:

- Optimierung des µC Quellcodes (mittels Profiling Laufzeiten berechnen und Wartezeiten anpassen, Prüfsummentest beim Empfangen von Zeichen)
- Portionierung des ClassPad-Programms als Add-In (C++)
- System netzwerkfähig gestalten (Lehrer-CP erfasst Messdaten und verteilt diese an die

Schüler → alle arbeiten unabhängig voneinander mit gleichen Datensätzen)

- andere Temperaturfühler nutzen, die genauer arbeiten/weniger störanfällig sind
- Entwicklung eines PC-Programms zur Auswertung (LabVIEW)
- Ausbau der Experimentierplatine und des Anwendungsbereiches (Physik, wissenschaftliches Praktikum, Informatik, Datenverarbeitung).

X. 3 Fazit

Das entwickelte Messsystem ist im Chemieunterricht gut einsetzbar. Es erreicht zwar nicht die höchste Genauigkeit, allerdings sollen in Schülerexperimenten auch nur Ansätze vermittelt werden und zeigen, dass etwaige Rechnungen durchaus adäquat sind. Exakte Messungen sind aufgrund der Umgebungsbedingungen ohnehin oft nicht möglich.

Ziel der Arbeit war es, den ClassPad als primäres Instrument der Mathematik stärker in andere Fächer zu integrieren und mathematische Zusammenhänge in Naturwissenschaften zu fördern (Aufstellen von Temperatur-Zeit-Funktionen, Interpretationen). Dies sollte zudem möglichst einfach geschehen, ohne dass Schulen großen Materialaufwand betreiben müssen. Da das Messerfassungssystem standardmäßig nur die Temperaturschnittstelle besitzt, ist es für den Chemieunterricht keinesfalls überdimensioniert. Sollte dennoch für die angesprochenen Le Chatelier-Experimente ein Drucksensor benötigt werden, ist dessen Anschluss möglich. Auch andere Fühler können angeschlossen werden. Die Programme sind quelloffen und damit auch veränderbar, um entsprechende Sensoren auszuwerten.

Letztendlich sind auch im CASIO-BASIC sehr einfache Grundbefehle verwendet worden, so dass eine Prüfung der Lauffähigkeit auf anderen OS-Versionen, wie sie in Kapitel VII.2 erwähnt wurde, entfallen kann.

In Hinblick auf diese Ziele ist das Ergebnis dieses Projektes ein voller Erfolg.

Abschließend sei zum angedachten Zeitaufwand von 286 Stunden gesagt, dass dieser deutlich überschritten wurde. Die gesamt investierte Arbeitszeit liegt bei ca. 700 Stunden. Dies kommt durch unvorhergesehene Programmierschwierigkeiten, Erlernen neuer Programme, viele Experimente, Recherchen und nicht zuletzt die schriftliche Dokumentation der genannten Punkte in entsprechender Form.

XI. Verzeichnisse

XI. 1 Allgemeines Abkürzungsverzeichnis

XI.1. 1 Abkürzungen für Anlagenverweise

A	Allgemeiner Anhang
EA	Evaluierungsauswertung
EAA	Akzeptanzanalyse
EV	Evaluierungsversuch
K	Korrespondenz
L	Literatur
Q	Quellen
QC	Quellcode
PAP	Programmablaufplan
PR	Protokoll
S	Sonstiges
SL	Schaltung, Schaltplan

XI.1. 2 Abkürzungsverzeichnis

A/D	Analog-Digital (Wandler)
ADC	Analog-Digital Converter (vgl. A/D)
ADCSRA	ADC Control and Status Register, Register für Einstellungen zur A/D-Wandlung
ADCH	AVR-Register mit High-Bytes des Ergebnisses einer A/D-Wandlung (vgl. ▹ A04**)**
ADCL	AVR-Register mit den Low-Bytes des Ergebnisses einer A/D-Wandlung (vgl. ▹ A04**)**
ADMUX	AVR-Register zur Kanalwahl für den A/D-Eingang und Setzen der Abtastrate
ADCW	AVR-Register, das ADCL und ADCH beinhaltet (vgl. ▹ A04)
AVR	Mikrocontroller von ATMEL
Bd	Baud (nach Jean-Maurice-Émile Baudot), Einheit für Übertragungsgeschwindigkeit serieller Schnittstellen in Zeichen/Sekunde
C, C++	Programmiersprachen
CAS	Computer Algebra System, z.B. nicht numerisches Lösen von Gleichungen
CISC	Complex Instruction Set Computing, Designphilosophie von Prozessoren (≠ RISC)
COM	serielle Schnittstelle am PC
CP	ClassPad
CPAD	von ClassPad – Analog/Digital, Bezeichnung für entwickelte Funkmesseinheit
D/A	Digital-Analog (Wandler)
el.	elektrisch, -er
EWA	Extremwert Aufgabe (*math.*), praxisbezogene Bestimmung von Extrema einer mathematischen Funktion
G	grafische Programmiersprache von National Instruments
GND	Ground, Masse

I²C serielles Bussystem

KTY Temperatursensormodell von Infineon

LM Operationsverstärkermodell

MPX Drucksensormodell von National Seminductor
µC allgemein für Mikrocontroller

OP, OV, OPV, OpAmp Operationsverstärker
OS Operating System, Betriebssystem/Firmware
PDIP standardisierte Einbauform el. Komponenten
PX Port X (vgl. ▷ A03)

R Widerstand, elektrischer
RFM12 Radio Frequency Module, 1: Empfangen, 2: Senden; Funkmodul
RISC Reduced Instruction Set Computing, Designphilosophie von Prozessoren (≠ CISC)
RS232 serielle Pegeldefinition

Therm. Thermometer
TTL serielle Pegeldefinition
TWI Two-Wire-Interface, ATMEL-eigenes I²C

UART serielles Sende- und Empfangsprotokoll

$\mathbf{V}_{CC}$ Versorgungsspannung
V_{REF} Referenzspannung
VI Virtual Instrument, Dateien von LabVIEW

Z Zeile, in Bezug auf Quelltextzeilen

XI. 2 Quellen- und Literaturverzeichnis

Quellenverzeichnis [Q]

Mikrocontroller

[Q1] ATmel: 8-bit Microcontroller – ATmega 44V/48V/168V.
http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf
Rev. 2545Q–AVR–06/09, Aufruf: 1.7.2009, 08:14

Temperaturfühler

[Q2] Infineon technologies: Silicon Temperature Sensors.
http://www.datasheetcatalog.org/datasheet/infineon/1-kt.pdf, 4.1.1999
Aufruf: 6.7.2009, 08:24

Operationsverstärker

[Q3] National Semiconductor: Low Power Dual Operational Amplifiers.
http://www.national.com/ds/LM/LM158.pdf, 10/2005
Aufruf: 22.7.2009, 14:09

[Q4] Semiconductor Components: Single Supply Dual Operational Amplifiers.
www.onsemi.com/pub_link/Collateral/LM358-D.PDF, 01/2008, Rev. 23
Aufruf: 25.7.2009, 12:18

integrierter Differenzverstärker

[Q5] Analog Devices: Low Cost, Low Power Instrumentation Amplifier – AD620.
http://www.datasheetcatalog.org/datasheet/analogdevices/105505445AD620_e.pdf
REV. E, Aufruf: 26.7.2009, 10:54

ClassPad300 PLUS

[Q6] CASIO: ClassPad 330 PLUS Bedienungsanleitung – Version 3.04
http://www.casio-europe.com/de/downloads/manuals/sgr/CP330_ver304_G.pdf,
Aufruf: 25.11.2009, 21:54

Drucksensor

[Q7] Freescale Semiconductor, Inc.: MPX4115A.
http://www.freescale.com/files/sensors/doc/data_sheet/MPX4115A.pdf
REV 5, 1/2009; Aufruf: 22.8.2009, 17:41

Literaturverzeichnis [L]

[L1] Sächsisches Staatsministerium für Kultus: Lehrplan Berufliches Gymnasium Chemie.
http://www.sachsen-macht-schule.de/apps/lehrplandb/downloads/lehrplaene/lp_bgy_
chemie_2007.pdf, Aufruf: 12.8.2009; 20:26

[L2] Wagner: 2. Informationsübertragung und -wandlung. - Heftermitschrift Technik Klasse 11,
BSZ Bau und Technik Dresden, 3.12.2007

[L3] Dr. K. Kreher, Dr. M. Krötzsch, Prof. Dr. H. A. Schneider u.a.: Physikalisches Praktikum.
BSB B. G. Teubner Verlagsgesellschaft, Leipzig 1985

[L4] Anhut, Martin, Höntsch, Johannes: Nutzung einer bidirektionalen Funkstrecke zur Datenübertragung am Computer Algebra System Casio ClassPad 300 PLUS. (Belegarbeit des Autors vom 23.2.2009)

[L5] Dr. sc. nat. Schmidt, Wolf-Dieter: Sensorschaltungstechnik.
Vogel Buchverlag, Würzburg 2002

[L6] Schwarz, Andreas: Operationsverstärker-Grundschaltungen.
http://www.mikrocontroller.net/articles/Operationsverstärker-Grundschaltungen
Aufruf: 29.7.2009; 11:00

[L7] Wellesz, John: classpad300, Processor speed. (Forum)
http://www.casiocalc.org/?showtopic=583, Aufruf: 12.8.2009; 22:40

[L8] Schwarz, Andreas: Checksumme berechnen?.
http://www.mikrocontroller.net/topic/95322, Aufruf: 12.9.2009; 20:40

[L9] CASIO Europe GmbH: „CASIO Schulrechner - Funktionen und Allgemeine Daten“
http://www.casio-schulrechner.de/de/produkte/grafikrechner/casrechner/classpad330/
Aufruf: 10.10.2009; 14:27

[L10] Prof. Dipl.-Math. Georgi, Wolfgang und Dipl.-Ing. Ergun, Metin: Einführung in LabVIEW.
4. Auflage, Carl Hanser Verlag, Leipzig 2009

[L11] Trampert, Wolfgang: Messen, Steuern und Regeln mit AVR-Mikrocontrollern.
1. Auflage, Franzis Verlag GmbH, Poing 2004

[L12] Prof. Dr. Kühn, Hartmut: Analoge Schaltungstechnik, Operationsverstärker (ideal).
http://www.htw-dresden.de/~hkuehn/Analoge_Schaltungstechnik/Vorlesung/
Schaltungstechnik-Vo-06a-OPV_ideal.pdf
Aufruf: 24.10.2010; 18:00

[L13] Free Software Foundation, Inc.: GNU GENERAL PUBLIC LICENSE Version 3, 3.7.07.
http://www.gnu.org/licenses/gpl-3.0.txt; Aufruf: 200.10.2009, 13:45

sonstige Korrespondenz [K]

[K1] E-Mail Kontakt zu CASIO: support_center@casio.de (Khalid Doulali): CAS-Produkte.
3.8.2009, 11:22

[K2] Interview mit Herrn Dr. Claus-Peter Schulz, Mathematiklehrer BSZ Bau und Technik,
Thema: „CAS im Unterricht“; 24.8.2009, 9:00 – 10:00 Uhr

[K3] E-Mail Kontakt zu Benedikt K. via mikrocontroller.net

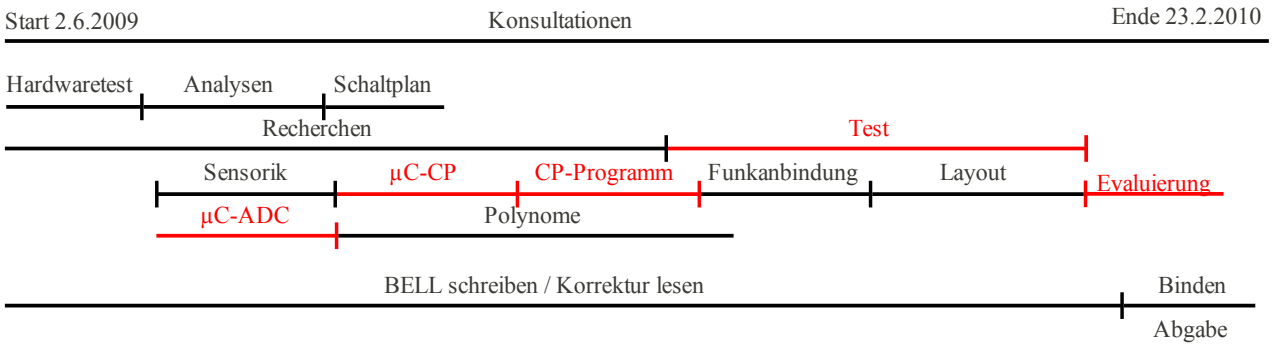
[K4] E-Mail Kontakt zu Prof. Dr. Paditz: Re: Leseprobe BELL.
19.11.2009, 17:00:17

XI. 3 Anlagenverzeichnis

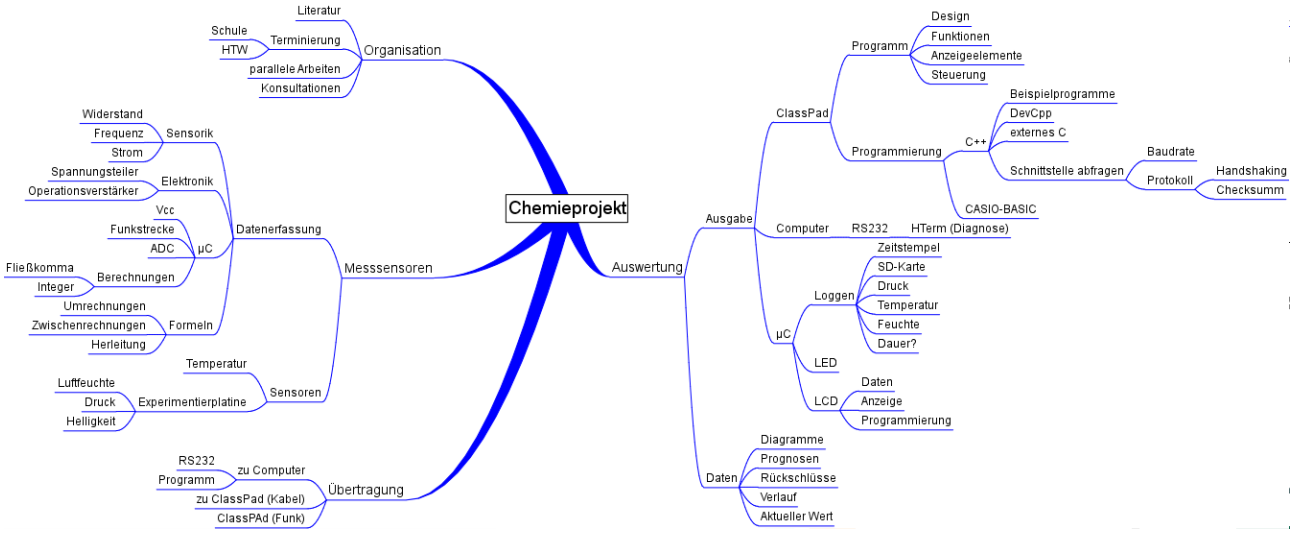
XII. Anhang

XII. 1 Allgemeiner Anhang

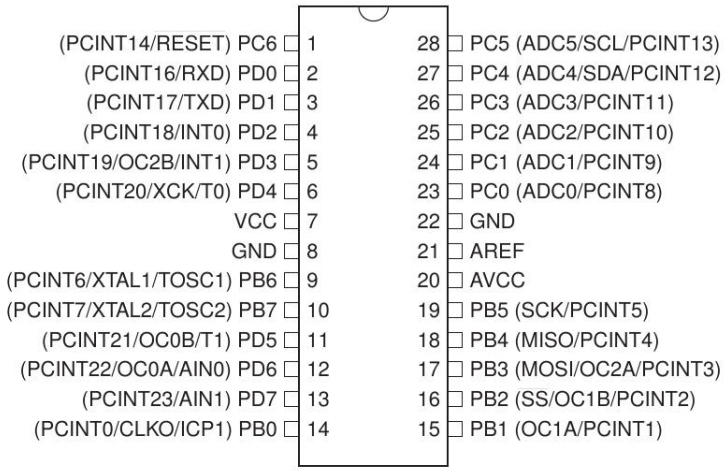
A01 – Projektplan



A02 – Mindmap



A03 – ATmega168 PDIP: Übersicht über die Pins

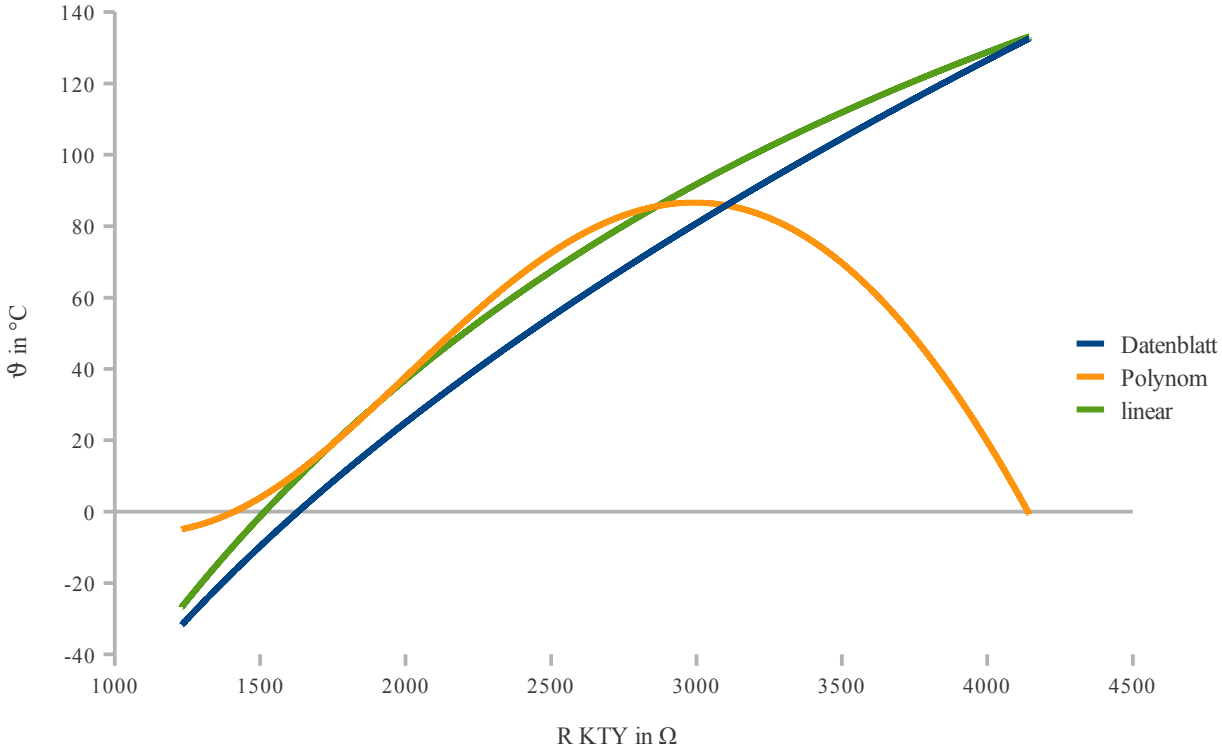


A04 – Die Ergebnisse der A/D-Wandlung in den Registern

Register	ADCW										
	ADCH		ADCL								
Wertigkeit	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Summe der Wertigkeiten
Dezimal	512	256	128	64	32	16	8	4	2	1	
Beispiele	0	0	0	0	0	0	0	0	0	0	= 0
	1	0	0	1	0	1	0	0	0	0	= 592
	1	1	1	1	1	1	1	1	1	1	= 1023

A05 – Diagramm der verschiedenen Temperaturberechnungsfunktionen

Die Bezeichnung „linear“ der dritten Kurve bezieht sich auf die Herleitung durch die Nutzung zweier Messpunkte. Damit wird $y = m \cdot x + n$ als äußere Funktion verwendet. Wegen $x = x(R_{KTY})$ als nichtlineare innere Funktion ist der Graph gekrümmt.



mit U_{ges} = 5 V, v = 2, U_1 = 1,5 V

zugehörige Formeln:

Datenblatt: $$\vartheta(x(R_{KTY})) = (25 + \frac{\sqrt{\alpha^2 - 4\beta + 4\beta \cdot \frac{R_{KTY}}{2000}} - \alpha}{2\beta}) \,^\circ C$$

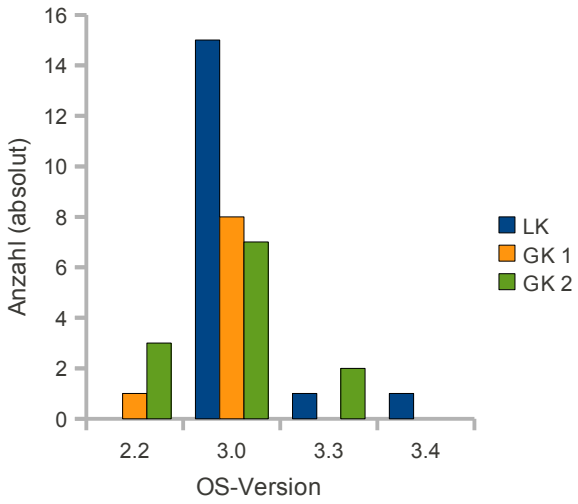
linear: $$\vartheta(x(R_{KTY})) = 0{,}265 \cdot x - 58{,}547$$

Polynom: $$\vartheta(x(R_{KTY})) = -2{,}53 \cdot 10^{-12} \cdot x^5 - 1{,}31 \cdot 10^{-12} \cdot x^4 + 1{,}37 \cdot 10^{-6} \cdot x^3 - 4{,}09 \cdot 10^{6} \cdot x^2 - 0{,}014 \cdot x - 5{,}47$$

wobei gilt: $x(R_{KTY}) = \frac{1024 \cdot v \cdot R_{KTY}}{R_{KTY} + 2200} - \frac{1024 \cdot v \cdot U_1}{U_{ges}}$ mit

U_{ges} - Gesamtspannung
v - OPV-Verstärkung
U_1 - OPV-Subtraktionsspannung

A06 – Diagramm über die Verteilung der OS-Versionen (BG07 BSZ Bau und Technik)[1]

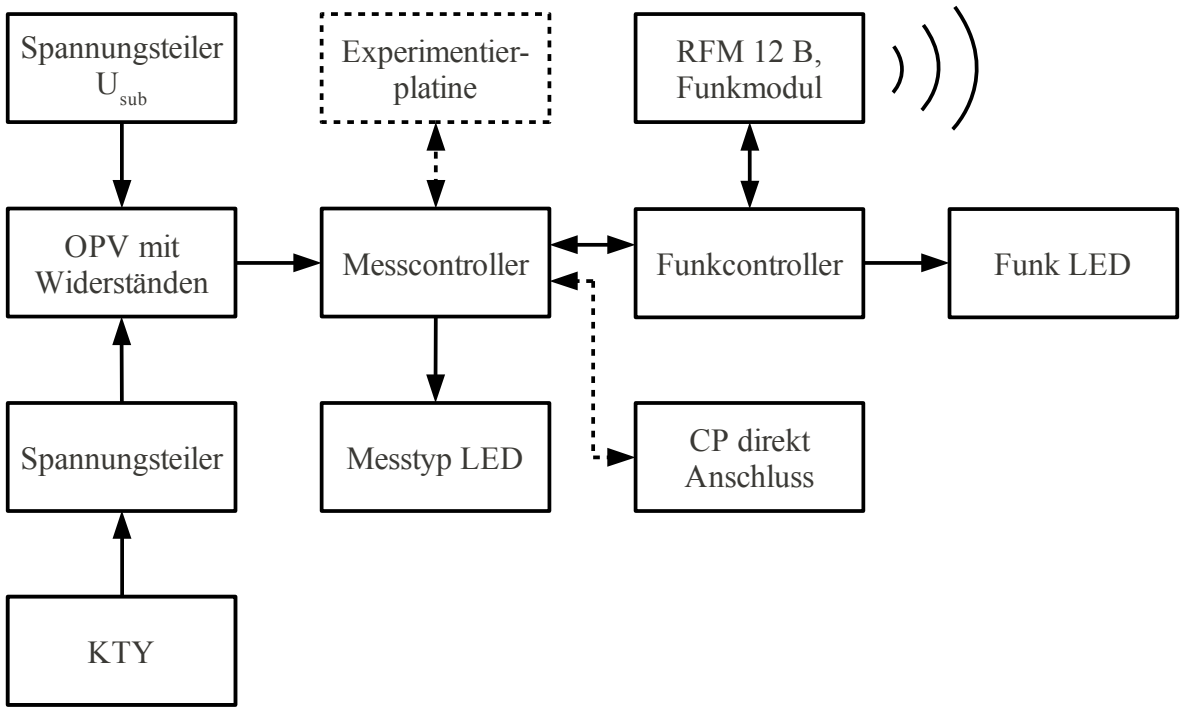


LK – Leistungskurs Mathematik
GK – Grundkurs Mathematik

Bei der Auslieferung 2007 war noch die Version 2.2 enthalten.

1 Erhebung vom 28.10.2009, Durchführung: Dr. Claus-Peter Schulz

A07 – Blockschaltbild zur Aufgabenverteilung



A08 – Druck erfassen und berechnen

Diese Anlage enthält theoretische Betrachtungen zum Anschluss eines Drucksensors, der nicht grundlegend Teil des Produktes sein soll, da er mit etwa 20 € nicht im Budget liegt.

Um den Druck zu ermitteln bedarf es weniger Bauteile und Rechnung, da der verwendete Sensor MPX4115A von Freescale Semiconductor eine Spannung in Abhängigkeit des Drucks ausgibt.

Die Schaltung ist aus dem Datenblatt entnommen und für Mikrocontroller bereits optimiert.

Für die Berechnung des Drucks aus dem Spannungswert fällt wie bei der Temperaturberechnung die Referenzspannung weg, wie folgende Rechnung zeigt:
die Formel nach dem Datenblatt: $V_{out} = V_S \cdot (0{,}009 \cdot p - 0{,}095) \pm Fehler$ (vgl. [Q7], S. 3, Abb. 3), V_S stellt dabei die Eingangsspannung dar. Da sie konstant bleibt, ist der Druck nur noch vom Ausgabewert abhängig:

$p(V_{out}) = \frac{100 \cdot V_{out}}{9 \cdot V_S} - \frac{95}{9}$. Ersetzt man nun V_{out} mit der Formel für die Strommessung am ADCW erhält man $p(V_{out}) = \frac{100 \cdot ADCW}{9 \cdot 1024} - \frac{95}{9}$; ($V_S = V_{REF}$). Der Ausgabewert ist in Kilopascal skaliert, für die übliche Einheit Hektopascal wird das Ergebnis mit 10 multipliziert.

XII. 2 Evaluierungsprotokolle und Akzeptanzanalysen

Folgend ist der Protokollteil angegeben, der allgemein für jede Evaluierung gilt, jedoch aus Platzgründen nicht bei den Tests angegeben ist

Ziel: Evaluierung der in der BELL entwickelten Hardware und Software durch Chemielehrer und Schüler im Rahmen zweier Experimente aus dem in Jahrgangsstufe 12 durchgeführten Praktikum „Energetische Betrachtungen bei chemischen Reaktionen“.

Vorüberlegung: Es wird ein Versuch ausgewählt, bei dem die Temperaturänderung stark ist (EV x.1, Erwärmung) und einer bei dem die Änderung relativ klein ist (EV x.2, Abkühlung). Damit soll die Gebrauchsfähigkeit in verschiedenen Szenarien geprüft werden.

Durchführung: Hardware, Software und eine Bedienungsanleitung werden den Testern vorgelegt. Sie haben (kurz) Zeit sich mit den Testgeräten vertraut zu machen. Danach werden die Evaluierungsversuche vorgelegt. Neben dem Versuch soll auch die Funktionalität des entwickelten Produktes ausgewertet werden. Auf Basis der Auswertung können weitere Anpassungen erfolgen. Zu jeder Akzeptanzanalyse (EAA) wird ein Protokoll angefertigt, das jede Evaluierung auswertet (EA).

Geordnet werden alle zur Evaluierung gehörigen Protokolle und Beiträge folgendermaßen:

Art des Protokolls **0** *Tester* **.** *zu Tester gehöriges Protokoll*

Beispiel: EV01.2 meint das zweite Protokoll (2) des Evaluierungsversuches (EV) des ersten Testers (1)

03.02.2010, 12:05	Evaluierung 1.1 : Versuch zur Evaluierung des in der BELL entwickelten Produktes (EV01.1))	Krug
Blatt 1/1		Dresden 301

Ziel: Untersuchen Sie das Temperaturänderungspotential von Kupfersulfat.

Vorüberlegung: $CuSO_4 + 5\ H_2O \underset{\text{endotherm}}{\overset{\text{exotherm}}{\rightleftharpoons}} CuSO_4 \cdot 5\ H_2O$

Während des Erhitzens wird sich das blaue Kupfersulfat weiß färben (Wasserentzug durch Wärmezufuhr), bei Wasserzugabe wieder blau.

Geräte/Chemikalien:

- Stativ
- Bunsenbrenner
- Reagenzglas
- (Spatel-) Löffel, Pipette
- Kupfersulfat ($CuSO_4$)
- Wasser (H_2O)
- ClassPad, Messgerät und Funkmodul

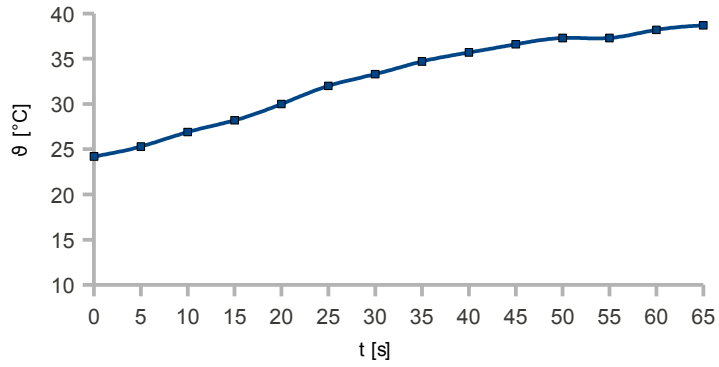
Durchführung:

- Reagenzglas schräg in Stativ einspannen
- einen Löffel $CuSO_4$ in RG geben
- vorsichtig erhitzen
- nach Abkühlung tropfenweise H_2O zugeben, Temperatur messen

Beobachtung:

t in s	0	5	10	15	20	25	30	35	40	45	50	55	60	65
ϑ in °C	24,2	25,3	26,9	28,2	30,0	32,0	33,3	34,7	35,7	36,6	37,3	37,3	38,2	38,7

Auswertung: Skizzieren Sie das Diagramm. Mit welcher Funktion könnte der Temperaturverlauf beschrieben werden? Geben Sie ein Beispiel an.



allgemeine Funktion:

$y = f(x) = e^x$

Beispiel für das Experiment:

$y = \vartheta(t) =$ ——

03.02.2010, 12:20	Evaluierung 1.2 : Versuch zur Evaluierung des in der BELL entwickelten Produktes (EV01.2))	Krug
Blatt 1/1		BSZ Zi 301

Ziel: Untersuchen Sie energetisch die Dissoziation von Kaliumnitrat. Berechnen Sie die freie Entalpie nach *Gibbs* mit den gegebenen Werten unter Standardbedingungen, interpretieren das Ergebnis und vergleichen Sie es mit dem Versuchsergebnis.

geg.: $\Delta_R H^0 = 35 \frac{kJ}{mol}$, $\Delta S^0 = 0{,}078 \frac{kJ}{K}$

Vorüberlegung: beim Lösevorgang Energie entziehen wird → endergone Reaktion → ΔG > 0 — Das Wasser wird sich schnell abkühlen, da das Kaliumnitrat dem Wasser

Geräte/Chemikalien:

- Reagenzglas
- Becherglas
- (Spatel-) Löffel, Pipette
- ClassPad und Messgerät, Funkmodule
- Kaliumnitrat (KNO_3)
- Wasser (H_2O)

Durchführung:

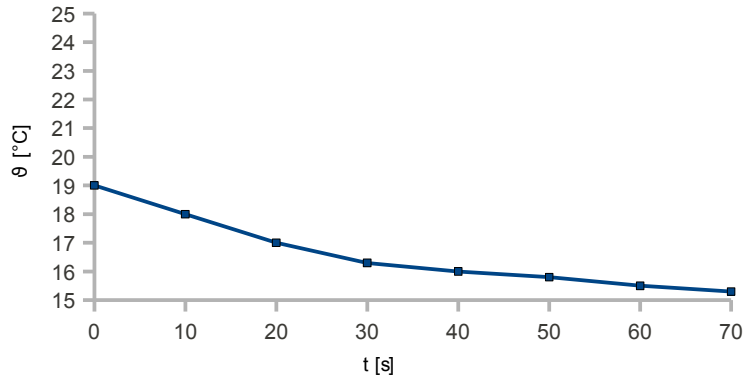
- in RG 2 cm hoch KNO_3 geben
- in Becherglas H_2O geben, Temperatur messen
- H_2O in RG geben, Temperatur messen

Beobachtung:

ϑ_{H2O} = 19,0 °C

t in s	0	10	20	30	40	50	60	70
ϑ in °C	19,0	18,0	17,0	16,3	16,0	15,8	15,5	15,3

Auswertung: Skizzieren Sie das Diagramm. Mit welcher Funktion könnte der Temperaturverlauf beschrieben werden? Geben Sie ein Beispiel an.



allgemeine Funktion:

$y = f(x) = e^{-x}$

Beispiel für das Experiment:

$y = \vartheta(t) =$ ———

Name: Krug

Akzeptanzanalyse zu den Evaluierungsversuchen 1.1 und 1.2 (EAA01)

Bitte jeweiliges ankreuzen, tragen Sie zudem bitte jeweils in die letzte Spalte Bewertungseinheiten von 0 bis 15 entsprechend der Bewertung der Jahrgangsstufen 12 und 13 ein.

1. Hardware:

Handlichkeit (Größe, Gewicht)

sehr sinnvoll	positiv	brauchbar X	negativ	störend	7 BE

Selbsterklärend (Elemente beschriftet und logisch angeordnet, Handhabung)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	10 BE

Informationen zum Messvorgang (Anzeige der Übertragung)

sehr sinnvoll X	positiv	brauchbar	negativ	störend	15 BE

Aufbau (Zeit/Komplikationen)

sehr schnell/keine	schnell/kleinere X	mittel/größere	lange/große	zu lange/zu viele	12 BE

2. Software:

Selbsterklärung (Menüs, Auswahl einer Aktion)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	12 BE

Übersichtlichkeit (Darstellung, Anordnung der Informationen)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	12 BE

Stabilität (Anzahl der Programmabstürze, Mängel bei der Kommunikation)

sehr stabil	kleine Mängel	zeitweise Fehler X	oft Fehler	nur Fehler	7 BE

Interaktion (Einstellmöglichkeiten)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	12 BE

Menüführung (Anzeige von Hilfen)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	12 BE

Auswerteoptionen (Messwertanzeige, Diagrammdarstellung)

sehr gut	gut	befriedigend X	mangelhaft	ungenügend	10 BE

Name: Krug

Einarbeitungszeit

keine	sehr klein X	mittel	groß	sehr groß	12 BE

Störanfälligkeit

Funkübertragung

keine	sehr klein	mittel X	groß	sehr (zu) groß	9 BE

Sensor

keine X	sehr klein	mittel	groß	sehr (zu) groß	15 BE

3. Handbuch

Anleitung allgemein

sehr nützlich	nützlich X	ausreichend	kompliziert	unverständlich	12 BE

Beschreibungen, Bilder, Erläuterungen

sehr nützlich	nützlich X	ausreichend	kompliziert	unverständlich	12 BE

Beschreibung zur Herleitung des Polynoms

sehr nützlich	nützlich	ausreichend	kompliziert	unverständlich	k.A. BE

Anhang

sehr nützlich X	nützlich	ausreichend	kompliziert	unverständlich	15 BE

Abschließende Bemerkungen vom Anwender (optional):

- Handbuch in Zweiteilung wäre günstiger
  (links Text, rechts Schnellanleitung)

05.02.2010, 21:00	Evaluierung 1.3 : Auswertung der Evaluierung und Akzeptanzanalyse von Frau Krug (EA01.3)	Johannes Höntsch
Blatt 1/1		zu Hause

Beobachtung: Aufgrund des kleinen Zeitfensters von 45 Minuten mussten zum Teil Erklärungen gegeben werden, welche aber auch im Handbuch standen. Vorher hatte die Testerin keine Informationen über Hard- und Software.

Auswertung: Die Testerin hatte selbst noch keine Erfahrungen mit dem ClassPad, dafür war die Bedingung des Programms (mit Bedienungsanleitung) aber hervorragend. Die Berechnungen wurden aufgrund des Zeitdrucks nicht mehr durchgeführt.

Punktabzüge gab es aus folgenden Gründen:

- Handlichkeit → etwas groß, hohes Gewicht
- Instabilität der Software → Anfangs mehrere Startversuche für Messvorgang nötig, dann aber keine Unterbrechungen mehr
- Auswerteoptionen → im Diagrammmessmenü kann nicht nach oben gescrollt werden, wenn das *Zurück?*-Fenster erscheint → bei längeren Messungen keine Anzeige aller Messwerte möglich → keine Regressionsmöglichkeit → keine speziellen Formeln
- Störung des Funks als Ursache für Instabilität der Software

Alle anderen Kategorien sind mit „gut“ bis „sehr gut“ bewertet wurden und bedürfen daher keiner Ursachenforschung und Änderung mehr.

Ergebnis: An einigen Punkten sind keine Änderungen mehr möglich (Gewicht), jedoch werden speziell aufgrund der Zusatzbemerkung folgende Änderungen vorgenommen:

- Handbuch in zweispaltigem Layout erstellen
- Ende der Diagrammmessung anders gestalten, so dass Messwerte noch angezeigt werden können; z.B. in Wertetabelle schreiben und dann Regressionen durchführen

Zudem wird die Instabilität des Programms untersucht und versucht zu reduzieren.

Die nächste Evaluierung wird mit entsprechenden Änderungen durchgeführt.

Nachtrag: Der Grund für die oft auftretende Fehlkommunikation ist bekannt und wurde behoben. Durch die Spannung von 6 V bei 4 AA-Batterien wurde der Controller kurz nach dem Messvorgang zurückgesetzt (wahrscheinlich, da an ADC0 mehr als 5 V anliegen konnten) und die Messung brach ab. Zur Lösung wurde eine Batterie entfernt und überbrückt. Alle anderen Änderungen wunden durchgeführt.

09.02.10, 14:00	Evaluierung 2.1 : Versuch zur Evaluierung des in der BELL entwickelten Produktes (EV02.1))	Dirk Bindszus
Blatt 1/1		J. Höntsch zu Hause

Ziel: Untersuchen Sie das Temperaturänderungspotential von Natriumhydroxid.

Vorüberlegung: $NaOH + H_2O \rightleftharpoons Na^+ + OH^- + H_2O$

Es wird Energie in Form von Wärme frei.

Geräte/Chemikalien:

- Becherglas
- (Spatel-) Löffel, Pipette
- ClassPad, Messgerät und Funkmodul
- Natriumhydroxid (NaOH)
- Wasser (H_2O)

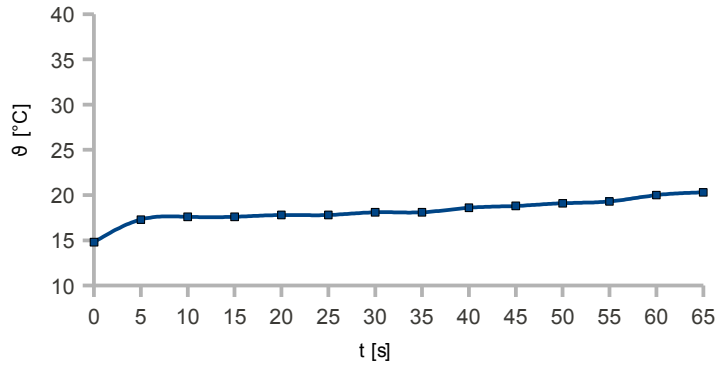
Durchführung:

- Wasser in Becherglas geben
- Temperaturfühler in Wasser
- NaOH hinzugeben und umrühren
- Temperaturverlauf aufzeichnen

Beobachtung:

t in s	0	5	10	15	20	25	30	35	40	45	50	55	60	65
ϑ in °C	14,8	17,3	17,6	17,6	17,8	17,8	18,1	18,1	18,6	18,8	19,1	19,3	20	20,3

Auswertung: Skizzieren Sie das Diagramm. Mit welcher Funktion könnte der Temperaturverlauf beschrieben werden? Geben Sie ein Beispiel an.



allgemeine Funktion:

$$y = f(x) = a \cdot e^{b \cdot x}$$

Beispiel für das Experiment:

$$y = \vartheta(t) = 16{,}295 \cdot e^{0{,}003 \cdot t}$$

09.02.2010, 14:30	Evaluierung 2.2: Versuch zur Evaluierung des in der BELL entwickelten Produktes (EV02.2))	Dirk Bindszus
Blatt 1/1		J. Höntsch zu Hause

Ziel: Untersuchen Sie energetisch die Dissoziation von Natriumchlorid. Berechnen Sie die freie Entalpie nach *Gibbs* mit den gegebenen Werten unter Standardbedingungen, interpretieren das Ergebnis und vergleichen Sie es mit dem Versuchsergebnis.

geg.: $\Delta_R H^0 = 35 \frac{kJ}{mol}$, $\Delta S^0 = 0{,}078 \frac{kJ}{K}$

Vorüberlegung: Das Wasser wird sich schnell abkühlen, da das Natriumchlorid dem Wasser beim Lösevorgang Energie entziehen wird → endergone Reaktion → $\Delta G > 0$

Geräte/Chemikalien:

– Becherglas
– (Spatel-) Löffel, Pipette
– ClassPad und Messgerät, Funkmodule
– Natriumchlorid (NaCl)
– Wasser (H_2O)

Durchführung:

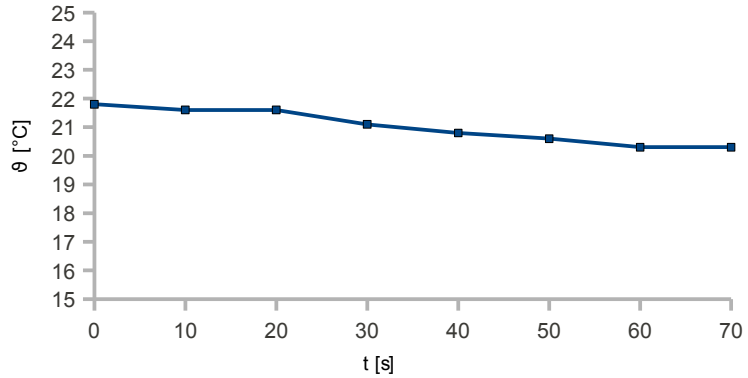
– in Becherglas H_2O geben, Temperatur messen
– NaCl zugeben und gut umrühren
– Temperaturverlauf messen

Beobachtung:

ϑ_{H2O} = 21,8 °C

t in s	0	10	20	30	40	50	60	70
ϑ in °C	21,8	21,6	21,6	21,1	20,8	20,6	20,3	20,3

Auswertung: Skizzieren Sie das Diagramm. Mit welcher Funktion könnte der Temperaturverlauf beschrieben werden? Geben Sie ein Beispiel an.



allgemeine Funktion:

$$y = f(x) = a \cdot e^{-b \cdot x}$$

Beispiel für das Experiment:

$$y = \vartheta(t) = 21{,}8 \cdot e^{-0{,}00133 \cdot t}$$

geg.: $\Delta_R H^0 = 35\,kJ/mol$ $\Delta S^0 = 0{,}078\,kJ/K$ ges.: $\Delta_R G\,[kJ/mol]$ Lsg.:

$$\Delta_R G = \Delta_R H^0 - T \cdot \Delta S^0$$
$$\Delta_R G = 35\,kJ/mol - 298\,K \cdot 0{,}078\,kJ/K$$
$$\Delta_R G = 11{,}76\,kJ/mol > 0$$

Da der Wert über der Nullgrenze liegt, bestätigt diese Formel das durchgeführte Experiment sowie die Vorüberlegungen. Das heißt, dass diese Reaktion endogen ist!

Name: Dirk Bindszus

Akzeptanzanalyse zu den Evaluierungsversuchen 2.1 und 2.2 (EAA02)

Bitte jeweiliges ankreuzen, tragen Sie zudem bitte jeweils in die letzte Spalte Bewertungseinheiten von 0 bis 15 entsprechend der Bewertung der Jahrgangsstufen 12 und 13 ein.

1. Hardware:

Handlichkeit (Größe, Gewicht)

sehr sinnvoll	positiv X	brauchbar	negativ	störend	11,5 BE

Selbsterklärend (Elemente beschriftet und logisch angeordnet, Handhabung)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	15 BE

Informationen zum Messvorgang (Anzeige der Übertragung)

sehr sinnvoll X	positiv	brauchbar	negativ	störend	15 BE

Aufbau (Zeit/Komplikationen)

sehr schnell/keine	schnell/kleinere X	mittel/größere	lange/große	zu lange/zu viele	12,5 BE

2. Software:

Selbsterklärung (Menüs, Auswahl einer Aktion)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	10 BE

Übersichtlichkeit (Darstellung, Anordnung der Informationen)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	11 BE

Stabilität (Anzahl der Programmabstürze, Mängel bei der Kommunikation)

sehr stabil X	kleine Mängel	zeitweise Fehler	oft Fehler	nur Fehler	15 BE

Interaktion (Einstellmöglichkeiten)

sehr gut	gut X	Befriedigend	mangelhaft	ungenügend	9 BE

Menüführung (Anzeige von Hilfen)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	11,5 BE

Auswerteoptionen (Messwertanzeige, Diagrammdarstellung)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	13 BE

Name: Dirk Bindszus

Einarbeitungszeit

keine	sehr klein	mittel X	groß	sehr groß	9,5 BE

Störanfälligkeit

Funkübertragung

keine X	sehr klein	mittel	groß	sehr (zu) groß	15 BE

Sensor

keine X	sehr klein	mittel	groß	sehr (zu) groß	15 BE

3. Handbuch

Anleitung allgemein

sehr nützlich	nützlich X	ausreichend	kompliziert X	unverständlich	10 BE

Beschreibungen, Bilder, Erläuterungen

sehr nützlich X	nützlich	ausreichend	kompliziert	unverständlich	14 BE

Beschreibung zur Herleitung des Polynoms

sehr nützlich	nützlich X	ausreichend	kompliziert X	unverständlich	10,5 BE

Anhang

sehr nützlich X	nützlich	ausreichend	kompliziert	unverständlich	15 BE

Abschließende Bemerkungen vom Anwender (optional):

- Änderungen von Parametern relativ aufwendig und irritierend.
- automatischer Sprung von Diagramm zum Statistikmenü wäre sinnvoll
- Ich (techn. Laie) hatte mit der Erklärung des 3-poligen Kabels so meine Schwierigkeiten
- als Prototyp sehr gelungen aber ausbaubar (sehr erwünscht)

09.02.2009, 18:00	Evaluierung 2.3 : Auswertung der Evaluierung und Akzeptanzanalyse von Dirk Bindzus) (EA02.3)	Johannes Höntsch
Blatt 1/1		zu Hause

Beobachtung: Die Versuche waren auf Anhieb möglich. Auch die anderen Chemikalien haben wie gewünscht reagiert.

Auswertung: Der Tester hat selbst einiges an Erfahrungen mit dem ClassPad (LK Wirtschaft). Die durch die erste Analyse festgestellten Probleme wurden inzwischen behoben und das Gerät zeigte gewünschte Funktionalität ohne Fehlfunktion.

Punktabzüge gab es aus folgenden Gründen:

- Handlichkeit → etwas groß
- Interaktion → soll nur eine Einstellung geändert werden, müssen auch die andern nicht zu ändernden Größen wieder eingegeben werden
- Anleitung → genaues (mehrfaches) Lesen erforderlich

Alle anderen Kategorien bedürfen keiner Ursachenforschung und Änderung mehr.

Ergebnis: Am Punkt „Handlichkeit“ ist keine Änderungen mehr möglich (Gewicht), jedoch werden speziell aufgrund der Zusatzbemerkung folgende Modifikationen vorgenommen/Überprüfungen erfolgen:

- Im Handbuch wird die Relativbeziehung „ , das dem ClassPad beiliegt“ eingefügt, um das zu verwendende Kabel zu spezifizieren
- Es ist zu überprüfen, inwiefern die Aufgaben im Statistikmenü auch aus dem Temperaturprogramm heraus erfolgen können.
- Es ist zu überprüfen, ob und wenn ja wie eine einzelne zu ändernde Diagrammeinstellung ausgewählt werden kann.

Nachtrag: Die Funktionen des Statistikmenüs wurden komplett in das Programm integriert. Hierzu wurde ein Unterprogramm entwickelt, dass nach der Diagrammmessung Optionen zur Regression anzeigt.
Um einzelne Änderungen für die Diagrammmessung einzugeben, wurde *wert* neu gestaltet und die bisherigen Bestandteile wurden in die Unterprogramme e_interv, e_zeit, e_untere und e_obere ausgelagert.

10.02.2010, 15:15	Evaluierung 3.1 : Versuch zur Evaluierung des in der BELL entwickelten Produktes (EV03.1))	Carl Richter
Blatt 1/1		J. Höntsch zu Hause

Ziel: Untersuchen Sie das Temperaturänderungspotential von Natriumhydroxid.

Vorüberlegung: $NaOH + H_2O \rightleftharpoons Na^+ + OH^- + H_2O$

Es wird Energie in Form von Wärme frei.

Geräte/Chemikalien:

- Becherglas
- (Spatel-) Löffel, Pipette
- ClassPad, Messgerät und Funkmodul
- Natriumhydroxid (NaOH)
- Wasser (H_2O)

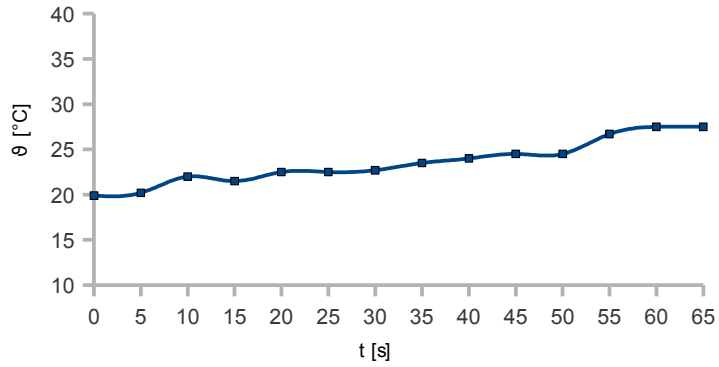
Durchführung:

- Wasser in Becherglas geben
- Temperaturfühler in Wasser
- NaOH hinzugeben und umrühren
- Temperaturverlauf aufzeichnen

Beobachtung:

t in s	0	5	10	15	20	25	30	35	40	45	50	55	60	65
ϑ in °C	19,9	20,2	22	21,5	22,5	22,5	22,7	23,5	24	24,5	24,5	26,7	27,5	27,5

Auswertung: Skizzieren Sie das Diagramm. Mit welcher Funktion könnte der Temperaturverlauf beschrieben werden? Geben Sie ein Beispiel an.



allgemeine Funktion:

$$y = f(x) = a \cdot e^{b \cdot x}$$

Beispiel für das Experiment:

$$y = \vartheta(t) = 20{,}04 \cdot e^{0{,}00488 \cdot t}$$

Wegen Störung des Funks musste das Experiment nochmal begonnen werden. Da bereits Natriumhydroxid im kalten Leitungswasser gelöst war liegt die Ausgangstemperatur bei 20 °C. Ungleichmäßiges Rühren fürhte zu zeitweiliger Abkühlung, da so keine Reaktion stattfindet.

10.02.2010, 16:30	Evaluierung 3.2 : Versuch zur Evaluierung des in der BELL entwickelten Produktes (EV03.2))	Carl Richter
Blatt 1/1		J. Höntsch zu Hause

Ziel: Untersuchen Sie energetisch die Dissoziation von Natriumchlorid. Berechnen Sie die freie Entalpie nach *Gibbs* mit den gegebenen Werten unter Standardbedingungen, interpretieren das Ergebnis und vergleichen Sie es mit dem Versuchsergebnis.

geg.: $\Delta_R H^0 = 35 \frac{kJ}{mol}$, $\Delta S^0 = 0{,}078 \frac{kJ}{K}$

Vorüberlegung: Das Wasser wird sich schnell abkühlen, da das Natriumchlorid dem Wasser beim Lösevorgang Energie entziehen wird → endergone Reaktion → $\Delta G > 0$

Geräte/Chemikalien:

– Becherglas
– (Spatel-) Löffel, Pipette
– ClassPad und Messgerät, Funkmodule
– Natriumchlorid (NaCl)
– Wasser (H_2O)

Durchführung:

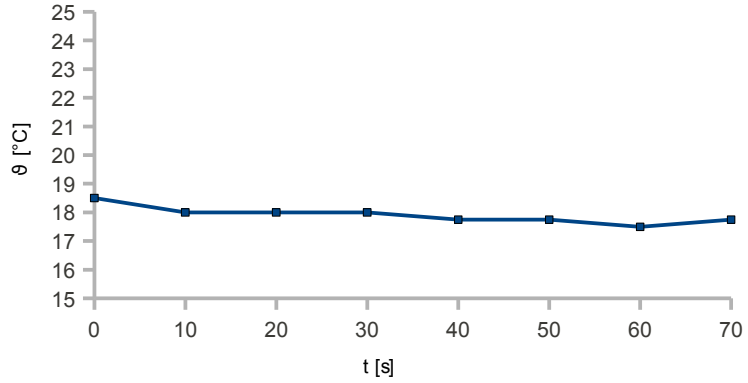
– in Becherglas H_2O geben, Temperatur messen
– NaCl zugeben und gut umrühren
– Temperaturverlauf messen

Beobachtung:

ϑ_{H2O} = 19,0 °C

t in s	0	10	20	30	40	50	60	70
ϑ in °C	18,5	18,0	18,0	18,0	17,8	17,8	17,5	17,8

Auswertung: Skizzieren Sie das Diagramm. Mit welcher Funktion könnte der Temperaturverlauf beschrieben werden? Geben Sie ein Beispiel an.



allgemeine Funktion:

$y = f(x) = a \cdot e^{-b \cdot x}$

Beispiel für das Experiment:

$y = \vartheta(t) = 18{,}27 \cdot e^{-0{,}000577 \cdot t}$

Lsg.: $\Delta_R G = \Delta_R H^0 - T \cdot \Delta S^0 = 35 \frac{kJ}{mol} - 298\,K \cdot 0{,}078 \frac{kJ}{K} = 11{,}76 \frac{kJ}{mol} > 0$

→ Reaktion ist endergon, d.h. es wird Energie gebraucht. Versuchsergebnis und Rechnung stimmen überein.

Name: Carl Richter

Akzeptanzanalyse zu den Evaluierungsversuchen 3.1 und 3.2 (EAA03)

Bitte jeweiliges ankreuzen, tragen Sie zudem bitte jeweils in die letzte Spalte Bewertungseinheiten von 0 bis 15 entsprechend der Bewertung der Jahrgangsstufen 12 und 13 ein.

1. Hardware:

Handlichkeit (Größe, Gewicht)

sehr sinnvoll	positiv X	brauchbar	negativ	störend	13 BE

Selbsterklärend (Elemente beschriftet und logisch angeordnet, Handhabung)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	12 BE

Informationen zum Messvorgang (Anzeige der Übertragung)

sehr sinnvoll X	positiv	brauchbar	negativ	störend	14 BE

Aufbau (Zeit/Komplikationen)

sehr schnell/keine X	schnell/kleinere	mittel/größere	lange/große	zu lange/zu viele	15 BE

2. Software:

Selbsterklärung (Menüs, Auswahl einer Aktion)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	13 BE

Übersichtlichkeit (Darstellung, Anordnung der Informationen)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	15 BE

Stabilität (keine Programmabstürze, Mängel bei der Kommunikation)

sehr stabil	kleine Mängel X	zeitweise Fehler	oft Fehler	nur Fehler	12 BE

Interaktion (Einstellmöglichkeiten)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	15 BE

Menüführung (Anzeige von Hilfen)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	14 BE

Auswerteoptionen (Messwertanzeige, Diagrammdarstellung)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	15BE

Name: Carl Richter

Einarbeitungszeit

keine	sehr klein X	mittel	groß	sehr groß	13 BE

Störanfälligkeit

Funkübertragung

keine	sehr klein X	mittel	groß	sehr (zu) groß	13 BE

Sensor

keine X	sehr klein	mittel	groß	sehr (zu) groß	15 BE

3. Handbuch

Anleitung allgemein

sehr nützlich X	nützlich	ausreichend	kompliziert	unverständlich	14 BE

Beschreibungen, Bilder, Erläuterungen

sehr nützlich X	nützlich	ausreichend	kompliziert	unverständlich	14 BE

Beschreibung zur Herleitung des Polynoms

sehr nützlich X	nützlich	ausreichend	kompliziert	unverständlich	14 BE

Anhang

sehr nützlich X	nützlich	ausreichend	kompliziert	unverständlich	15 BE

Abschließende Bemerkungen vom Anwender (optional):

- Handlichkeit → etwas groß
- eine Störung → Experiment musste neu begonnen werden

10.02.2009, 19:00	Evaluierung 3.3 : Auswertung der Evaluierung und Akzeptanzanalyse von Carl Richter (EA03))	Johannes Höntsch
Blatt 1/1		zu Hause

Beobachtung: Beim ersten Versuch gab es einen Fehler und die Messung musste erneut begonnen werden.

Auswertung: Der Tester hat selbst einiges an Erfahrungen mit dem ClassPad (LK Mathe, Datenverarbeitung). Die automatische Regression hat problemlos funktioniert und bleibt damit Bestandteil des Programms ohne Änderungen.

Punktabzüge gab es aus folgenden Gründen:

- Handlichkeit → etwas groß
- Stabilität → im Zusammenhang mit einer Kommunikationsstörung

Die Störung kann als Ausnahme angesehen werden, da es in vielen (nicht protokollierten) Testläufen seither noch keine Fehlfunkton gab.

Ergebnis: Aus dieser Evaluierung lassen sich keine Verbesserungsmöglichkeiten ableiten.

13.02.2010, 14:00	Protokoll 4.1: Versuch zur Evaluierung des in der BELL entwickelten Produktes (EV04.1)	André Schubert
Blatt 1/1		J. Höntsch zu Hause

Ziel: Untersuchen Sie das Temperaturänderungspotential von Natriumhydroxid.

Vorüberlegung: $NaOH + H_2O \rightleftharpoons Na^+ + OH^- + H_2O$

Es wird Energie in Form von Wärme frei.

Geräte/Chemikalien:

- Becherglas
- (Spatel-) Löffel, Pipette
- ClassPad, Messgerät und Funkmodul
- Natriumhydroxid (NaOH)
- Wasser (H_2O)

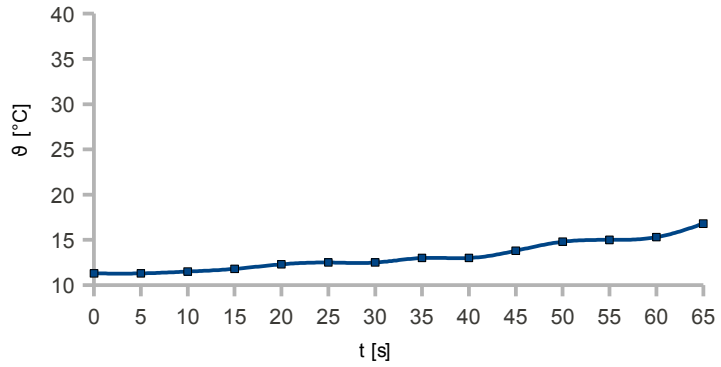
Durchführung:

- Wasser in Becherglas geben
- Temperaturfühler in Wasser
- NaOH hinzugeben und umrühren
- Temperaturverlauf aufzeichnen

Beobachtung:

t in s	0	5	10	15	20	25	30	35	40	45	50	55	60	65
ϑ in °C	11,3	11,3	11,5	11,8	12,3	12,5	12,5	13	13	13,8	14,8	15	15,3	16,8

Auswertung: Skizzieren Sie das Diagramm. Mit welcher Funktion könnte der Temperaturverlauf beschrieben werden? Geben Sie ein Beispiel an.



allgemeine Funktion:

$$y = f(x) = a \cdot e^{x}$$

Beispiel für das Experiment:

$$y = \vartheta(t) = 10{,}856 \cdot e^{0{,}00579 \cdot t}$$

13.02.2010, 14:30	Protokoll 4.2: Versuch zur Evaluierung des in der BELL entwickelten Produktes	André Schubert
Blatt 1/1		J. Höntsch zu Hause

Ziel: Untersuchen Sie energetisch die Dissoziation von Natriumchlorid. Berechnen Sie die freie Entalpie nach *Gibbs* mit den gegebenen Werten unter Standardbedingungen, interpretieren das Ergebnis und vergleichen Sie es mit dem Versuchsergebnis.

geg.: $\Delta_R H^0 = 35 \frac{kJ}{mol}$, $\Delta S^0 = 0{,}078 \frac{kJ}{K}$

Vorüberlegung: Das Wasser wird sich schnell abkühlen, da das Natriumchlorid dem Wasser beim Lösevorgang Energie entziehen wird → endergone Reaktion → $\Delta G > 0$

Geräte/Chemikalien:

- Becherglas
- (Spatel-) Löffel, Pipette
- ClassPad und Messgerät, Funkmodule
- Natriumchlorid (NaCl)
- Wasser (H_2O)

Durchführung:

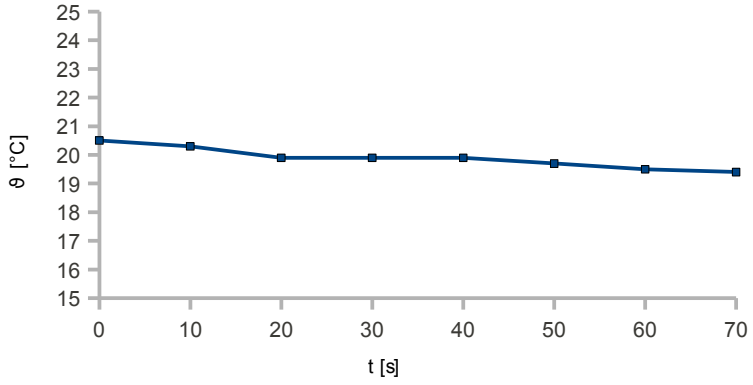
- in Becherglas H_2O geben, Temperatur messen
- NaCl zugeben und gut umrühren
- Temperaturverlauf messen

Beobachtung:

ϑ_{H2O} = 20,5 °C

t in s	0	10	20	30	40	50	60	70
ϑ in °C	20,5	20,3	19,9	19,9	19,9	19,7	19,5	19,4

Auswertung: Skizzieren Sie das Diagramm. Mit welcher Funktion könnte der Temperaturverlauf beschrieben werden? Geben Sie ein Beispiel an.



allgemeine Funktion:

$$y = f(x) = a \cdot e^{-b \cdot x}$$

Beispiel für das Experiment:

$$y = \vartheta(t) = 20{,}411 \cdot e^{-0{,}000684 \cdot t}$$

Lsg.: Standardbedingungen → $T = 298\ K$

$$\Delta_R G = \Delta_R H^0 - T \cdot \Delta S^0 = 11{,}756 \frac{kJ}{mol} > 0 \Rightarrow endergon$$

q.e.d.

Name: André Schubert

Akzeptanzanalyse zu den Evaluierungsversuchen 4.1 und 4.2 (EAA04)

Bitte jeweiliges ankreuzen, tragen Sie zudem bitte jeweils in die letzte Spalte Bewertungseinheiten von 0 bis 15 entsprechend der Bewertung der Jahrgangsstufen 12 und 13 ein.

1. Hardware:

Handlichkeit (Größe, Gewicht)

sehr sinnvoll X	positiv	brauchbar	negativ	störend	14 BE

Selbsterklärend (Elemente beschriftet und logisch angeordnet, Handhabung)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	12 BE

Informationen zum Messvorgang (Anzeige der Übertragung)

sehr sinnvoll	positiv X	brauchbar	negativ	störend	12 BE

Aufbau (Zeit/Komplikationen)

sehr schnell/keine X	schnell/kleinere	mittel/größere	lange/große	zu lange/zu viele	15 BE

2. Software:

Selbsterklärung (Menüs, Auswahl einer Aktion)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	11 BE

Übersichtlichkeit (Darstellung, Anordnung der Informationen)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	11 BE

Stabilität (Anzahl der Programmabstürze, Mängel bei der Kommunikation)

sehr stabil	kleine Mängel X	zeitweise Fehler	oft Fehler	nur Fehler	12 BE

Interaktion (Einstellmöglichkeiten)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	15 BE

Menüführung (Anzeige von Hilfen)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	14 BE

Auswerteoptionen (Messwertanzeige, Diagrammdarstellung)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	15 BE

Name: André Schubert

Einarbeitungszeit

keine	sehr klein X	mittel	groß	sehr groß	13 BE

Störanfälligkeit

Funkübertragung

keine X	sehr klein	mittel	groß	sehr (zu) groß	15 BE

Sensor

keine X	sehr klein	mittel	groß	sehr (zu) groß	14 BE

3. Handbuch

Anleitung allgemein

sehr nützlich X	nützlich	ausreichend	kompliziert	unverständlich	15 BE

Beschreibungen, Bilder, Erläuterungen

sehr nützlich	nützlich X	ausreichend	kompliziert	unverständlich	15 BE

Beschreibung zur Herleitung des Polynoms

sehr nützlich	nützlich X	ausreichend	kompliziert	unverständlich	14 BE

Anhang

sehr nützlich X	nützlich	ausreichend	kompliziert	unverständlich	15 BE

Abschließende Bemerkungen vom Anwender (optional):

- Anordnung der Auswahlmenüs zusammenfassen in einem Block zur besseren Übersicht

Bsp: (Anzeige)

(Menüauswahl)

13.02.2010	Evaluierung 4.3: Auswertung der Evaluierung und Akzeptanzanalyse von André Schubert (EA04)	Johannes Höntsch
Blatt 1/1		zu Hause

Beobachtung: Es gab keine nennenswerten Vorkommnisse.

Auswertung: Der Tester hatte selbst viel Erfahrung mit dem ClassPad (LK Mathe, Bautechnik). Entsprechend gab es bei dem Test keine Bedienprobleme.

Punktabzüge gab es aus folgenden Gründen:

- Selbsterklärung/Übersichtlichkeit → Anordnung von Eingabe- und Informationsbereichen

Alle anderen Kategorien sind mit „gut“ bis „sehr gut“ bewertet wurden und bedürfen daher keiner Ursachenforschung und Änderung mehr.

Ergebnis: Im Hauptanzeigeprogramm *temp1* und dem Einstellungsmenü *einst* werden die Bereiche für Eingabemöglichkeiten und Informationsanzeige in Blöcken zusammengefasst.

16.02.2010, 13:35	Protokoll 5.1: Versuch zur Evaluierung des in der BELL entwickelten Produktes (EV05.1)	Alrik Künne
Blatt 1/1		A. Künne zu Hause

Ziel: Untersuchen Sie das Temperaturänderungspotential von Natriumhydroxid.

Vorüberlegung: $NaOH + H_2O \rightleftharpoons Na^+ + OH^- + H_2O$

Es wird Energie in Form von Wärme frei.

Geräte/Chemikalien:

- Becherglas
- (Spatel-) Löffel, Pipette
- ClassPad, Messgerät und Funkmodul
- Natriumhydroxid (NaOH)
- Wasser (H_2O)

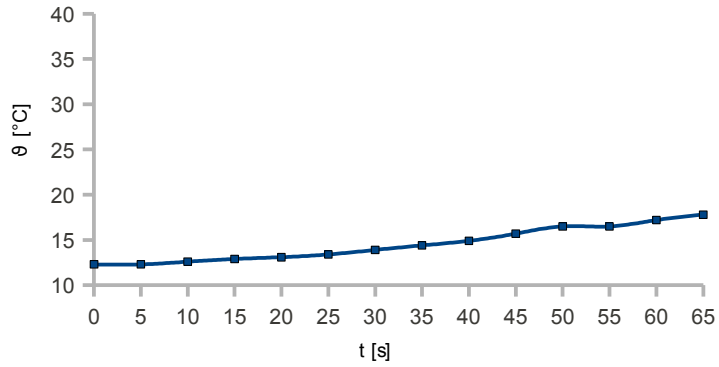
Durchführung:

- Wasser in Becherglas geben
- Temperaturfühler in Wasser
- NaOH hinzugeben und umrühren
- Temperaturverlauf aufzeichnen

Beobachtung:

t in s	0	5	10	15	20	25	30	35	40	45	50	55	60	65
ϑ in °C	12,3	12,3	12,6	12,9	13,1	13,4	13,9	14,4	14,9	15,7	16,5	16,5	17,2	17,8

Auswertung: Skizzieren Sie das Diagramm. Mit welcher Funktion könnte der Temperaturverlauf beschrieben werden? Geben Sie ein Beispiel an.



allgemeine Funktion:

$$y = f(x) = a \cdot e^{b \cdot x}$$

Beispiel für das Experiment:

$$y = \vartheta(t) = 11{,}837 \cdot e^{0{,}006074 \cdot t}$$

16.02.2010, 13:50	Protokoll 5.2: Versuch zur Evaluierung des in der BELL entwickelten Produktes	Alrik Künne
Blatt 1/1		A. Künne zu Hause

Ziel: Untersuchen Sie energetisch die Dissoziation von Natriumchlorid. Berechnen Sie die freie Entalpie nach *Gibbs* mit den gegebenen Werten unter Standardbedingungen, interpretieren das Ergebnis und vergleichen Sie es mit dem Versuchsergebnis.

geg.: $\Delta_R H^0 = 35 \frac{kJ}{mol}$, $\Delta S^0 = 0{,}078 \frac{kJ}{K}$

Vorüberlegung: Das Wasser wird sich schnell abkühlen, da das Natriumchlorid dem Wasser beim Lösevorgang Energie entziehen wird → endergone Reaktion → $\Delta G > 0$

Geräte/Chemikalien:

– Becherglas
– (Spatel-) Löffel, Pipette
– ClassPad und Messgerät, Funkmodule
– Natriumchlorid (NaCl)
– Wasser (H_2O)

Durchführung:

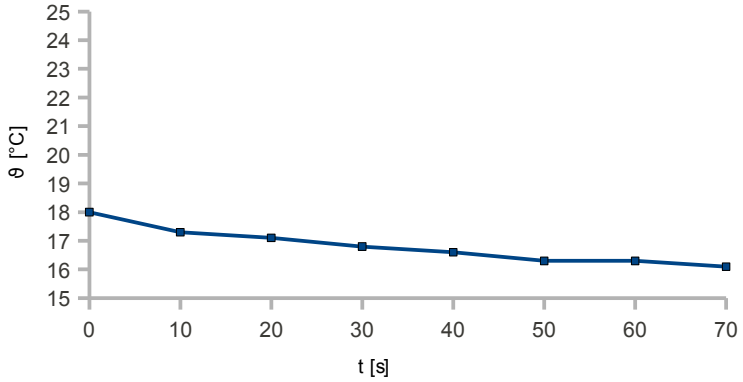
– in Becherglas H_2O geben, Temperatur messen
– NaCl zugeben und gut umrühren
– Temperaturverlauf messen

Beobachtung:

$\vartheta_{H2O} =$ 18,0 °C

t in s	0	10	20	30	40	50	60	70
ϑ in °C	18,0	17,3	17,1	16,8	16,6	16,3	16,3	16,1

Auswertung: Skizzieren Sie das Diagramm. Mit welcher Funktion könnte der Temperaturverlauf beschrieben werden? Geben Sie ein Beispiel an.



allgemeine Funktion:

$$y = f(x) = a \cdot e^{-b \cdot x}$$

Beispiel für das Experiment:

$$y = \vartheta(t) = 17{,}689 \cdot e^{-0{,}001455 \cdot t}$$

Lsg.: Standardbedingungen → $T = 298\ K$

$$\Delta_R G = \Delta_R H^0 - T \cdot \Delta S^0 = 11{,}756 \frac{kJ}{mol} > 0$$

Rechnung bestätigt Vorüberlegung und Experiment.

Name: Alrik Künne

Akzeptanzanalyse zu den Evaluierungsversuchen 5.1 und 5.2 (EAA05)

Bitte jeweiliges ankreuzen, tragen Sie zudem bitte jeweils in die letzte Spalte Bewertungseinheiten von 0 bis 15 entsprechend der Bewertung der Jahrgangsstufen 12 und 13 ein.

1. Hardware:

Handlichkeit (Größe, Gewicht)

sehr sinnvoll X	positiv	brauchbar	negativ	störend	14 BE

Selbsterklärend (Elemente beschriftet und logisch angeordnet, Handhabung)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	15 BE

Informationen zum Messvorgang (Anzeige der Übertragung)

sehr sinnvoll X	positiv	brauchbar	negativ	störend	15 BE

Aufbau (Zeit/Komplikationen)

sehr schnell/keine X	schnell/kleinere	mittel/größere	lange/große	zu lange/zu viele	14 BE

2. Software:

Selbsterklärung (Menüs, Auswahl einer Aktion)

sehr gut	gut	befriedigend X	mangelhaft	ungenügend	10 BE

Übersichtlichkeit (Darstellung, Anordnung der Informationen)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	12 BE

Stabilität (Anzahl der Programmabstürze, Mängel bei der Kommunikation)

sehr stabil	kleine Mängel X	zeitweise Fehler	oft Fehler	nur Fehler	9 BE

Interaktion (Einstellmöglichkeiten)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	13 BE

Menüführung (Anzeige von Hilfen)

sehr gut	gut X	befriedigend	mangelhaft	ungenügend	11 BE

Auswerteoptionen (Messwertanzeige, Diagrammdarstellung)

sehr gut X	gut	befriedigend	mangelhaft	ungenügend	15 BE

Name: Alrik Künne

Einarbeitungszeit

keine	sehr klein X	mittel	groß	sehr groß	13 BE

Störanfälligkeit

Funkübertragung

keine	sehr klein X	mittel	groß	sehr (zu) groß	12 BE

Sensor

keine X	sehr klein	mittel	groß	sehr (zu) groß	15 BE

3. Handbuch

Anleitung allgemein

sehr nützlich X	nützlich	ausreichend	kompliziert	unverständlich	15 BE

Beschreibungen, Bilder, Erläuterungen

sehr nützlich	nützlich X	ausreichend	kompliziert	unverständlich	13 BE

Beschreibung zur Herleitung des Polynoms

sehr nützlich	nützlich X	ausreichend	kompliziert	unverständlich	15 BE

Anhang

sehr nützlich	nützlich X	ausreichend	kompliziert	unverständlich	12 BE

Abschließende Bemerkungen vom Anwender (optional):

positiv:

- eingebautes Regressionsmenü
- Setup
- Menüführung

negativ:

- für Unwissenden ist die Funktion des Polynoms nicht klar
- Einheiten in der Menüführung angeben

16.02.2009, 21:00	Evaluierung 5.3 : Auswertung der Evaluierung und Akzeptanzanalyse von Alrik Künne) (EA05)	Johannes Höntsch
Blatt 1/1		zu Hause

Beobachtung: Die Versuche waren auf Anhieb möglich. Beim Ausprobieren gab es einen Programmabsturz wegen Fehlübertragung.

Auswertung: Der Tester hat selbst einiges an Erfahrungen mit dem ClassPad (LK Mathe). Die Test wurden in sehr kurzer Zeit durchgeführt. Die ermittelten Messwerte sind für die Experimente sehr repräsentativ, was zeigt, dass Hardware und Software bereits sehr gut funktionieren.

Punktabzüge gab es aus folgenden Gründen:

- Stabilität → ein Programmabsturz beim Vertrautmachen mit den Komponenten

Alle anderen Kategorien bedürfen keiner Ursachenforschung und Änderung mehr.

Ergebnis: Folgende Änderungen werden übernommen:

- beim Messintervall wird die Einheit hinzugefügt
- im Handbuch wird durch grobe Erläuterung der Funktionsweise die Rolle des Polynoms erläutert

XII. 3 Quellcodes

QC01 – A/D-Wandler Werte kontinuierlich per UART ausgeben

```
1  /* Programm fuer Ansteuerung des ADC eines ATmega44/88/168
2  ** - Init des UART nach P. Fleury (38,4 kBd)
3  ** - Init der AD-Register: Aref = Avcc, Teiler:64, Quelle:ADC0
4  ** - eine Probemessung zum aufwaermen
5  ** - Messen (aus 16 Mittelwert) aller 2 Sekunden und Ausgabe via UART
6  **
7  **  Johannes Hoetsch, v5.0, 18.7.2009
8  */
9
10 #include <stdlib.h>
11 #include <avr/io.h>
12 #include <avr/interrupt.h>
13 #include <avr/signal.h>
14 #include <avr/pgmspace.h>
15 #include <util/delay.h>
16
17 #include "uart.h"
18
19 #define UART_BAUD_RATE 38400 //ClassPad Bd
20
21 int main(void)
22  {
23
24 /*Variablendeklaration*/
25   uint16_t e=0; //Ergebnisvariable
26   uint8_t i=0; //Zaelvariable
27   volatile char s[16]; //Ergebnisstring
28
29 /*UART-Init und Test*/
30   uart_init( UART_BAUD_SELECT_DOUBLE_SPEED(UART_BAUD_RATE,F_CPU) );
31   sei();
32   _delay_ms(100); //0,1s warten, bis Init abgeschlossen
33   uart_puts("UART OK\n");
34
35 /* A/D-Register setzen */
36    ADMUX = 0; //von ADC0 lesen
37    ADMUX |= (1<<REFS0); // A/D-Referenzbits auf AVCC setzen
38    ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1); // A/D-Wandler aktiv
39
40 /*Dummy-Readout */
41    ADCSRA |= (1<<ADSC); // A/D-Wandlung starten
42    while (ADCSRA & (1<<ADSC)) //solange Konvertierung mache nichts
43     {;}
44    e = ADCW; //Wert nach e schreiben
45
46 /*Messschleife*/
47  for(;;)
48   {
49     e = 0; //Ergebnis zuruecksetzen
50
51     for(i=0; i<16; i++) //16x messen
52      {
53        ADCSRA |= (1<<ADSC); // A/D-Wandlung starten
54        while (ADCSRA & (1<<ADSC)) //Solange Konvertierung mache nichts
55         {;}
56        e = e + ADCW; //Messungen aufsummieren
57      }
58     e = (e>>4); //Mittelwert aus 16 Messungen bilden
```

```
59
60      itoa(e,s,10); //Ergebnis als Zeichenkette um per UART auszugeben
61      uart_puts(s); //ADCW per UART ausgeben
62
63      _delay_ms(2000); //2s warten
64    }
65  }
```

QC02 – Berechnung der Temperatur nach Datenblatt (für PC)

```
 1 /*Testprogramm um fuer gegebenen ADCW Temperatur zu berechnen
 2 ** - gibt Rx, kt und T aus
 3 ** - Fehlermeldung bei unter-/ueberschrittenem Wertebereich
 4 ** - Ausgabe das zulaessigen Wertebereiches
 5 **
 6 **  Johannes Hoentsch, v1.10, 21.7.2009
 7 */
 8
 9 #include <stdio.h>
10 #include <math.h>
11 int main()
12 {
13  double adcw = 0; //ADCW-Wert
14  double r_1 = 0;        //Shunt-Wert
15  int min = 0;           //Wertebereichsminimum
16  int max = 0;           //Wertebereichsmaximum
17  double r_x = 0;        //Widerstand des Sensors
18  double kt = 0;
19  double t = 0;
20  double a = (7.88/1000);
21  double b = (1.937/100000);
22
23  for(;;)
24  {
25     start:                    //Startmarke fuer Wertebereichspruefung
26
27     printf("\nR1=");          //Shunt eingeben
28     scanf("%lf", &r_1);
29
30     min = ((r_1 * 1024)/(r_1 + 4144));    //Wertebereichsminimum
31     max = ((r_1 * 1024)/(r_1 + 1230));   //Wertebereichsmaximum
32
33     printf("ADCW %d < ADCW < %d", min, max);    //ADCW eingeben
34     scanf("%lf", &adcw);
35
36 /*Pruefung des Wertebereiches*/
37     if(adcw < min + 1 )
38      { printf("Wertebereich unterschritten! (min=%d)\n\n", min+1); goto
          start; }
39
40     if(adcw > max-1)
41      { printf("Wertebereich ueberschritten! (max=%d)\n\n", max-1); goto
          start; }
42
43 /*Berechnungen*/
44     r_x = ((r_1* 1024)/adcw) -r_1;  //Rx nach Spannungsteilerregel berechnen
45     kt = r_x/2000;                  //Verhaeltnis von Widerstand zu 25 °C
46     t = (( sqrt( a*a - (4 * b) + (4 * b * kt )  ) - a)  / ( 2 * b ) ) + 25;
47
48 /*Ergebnisausgaben*/
49     printf("Rx = %lf\t", r_x);      //berechneten Rx ausgeben
```

```
50     printf("kt = %lf\t", kt); //berechnetes Verhältnis ausgeben
51     printf("T = %lf °C\n\n", t);    //berechnete Temperatur ausgeben
52  }
53  return 0;
54 }
```

QC03 – Heronverfahren (PC)

```
 1 /*Wurzel einer Zahl berechnen (Heronverfahren)
 2 ** - Eingabe des Wurzelwertes
 3 ** - Berechnung in Unterprogramm "wurzel(x)"
 4 ** - formatierte Ausgabe des Ergebnisses
 5 **
 6 **   Johannes Hoentsch, v2.10, 24.9.2009
 7 */
 8
 9 #include <stdio.h>
10
11 double a = 25; //Startwert der Iteration
12
13 int main()
14  {
15
16   double x = 0;          //Radikant
17   printf("Radikant=");  //Radikant eingeben
18   scanf("%lf", &x);     //Eingabe nach x schreiben
19
20   wurzel(x);            //Funktionsaufruf
21   printf("wurzel %lf = %f\t", x,a);
22
23   return 0;
24  }
25
26
27 int wurzel (double x)
28  {
29   int i = 0; //Schleifenindex
30
31   for(i=0; i <6; i++)
32   {
33    a = (a + (x/a))/2;
34   }
35   return a;
36  }
```

QC04 – ClassPad-Programm zum Senden eines Strings

```
1 InputStr s        //Zeichen eingeben und als String s definieren
2 OpenComPort38k    //Öffne die Schnittstelle (38 kBd)
3 SendVar38k s      //Sende den String s (mit 38 kBd)
4 CloseComPort38k   //Schließe die Schnittstelle (38 kBd)
```

QC05 – Das Unterprogramm casio.c

Header-Datei:

```
/*************************************************************
** Funktionen:    Sendet bis zu 4 Zahlen nach ClassPad Protokoll
**                Empfängt 3 Zeichen nach ClassPad Protokoll
** Datum:   20.10.2009, v1.1.0-1
** gcc Version: 4.1.3 20080612 (prerelease) (SUSE Linux)
** Lizenz:  GNU GENERAL PUBLIC LICENSE
**
** Copyright (C) 2009,  Johannes Höntsch, j.hoentsch@gmx.de
**
**    GNU-Auszug zur Geltendmachung der Lizenz
**
*************************************************************/

#include "uart.h"

#define UART_BAUD_RATE 38400

void cp_send(int wert);
int cp_get(void);
```

C-Datei:

```
/***********************************************
** Funktionen:    Sendet bis zu 4 Zahlen nach ClassPad Protokoll
**          Empfängt 3 Zeichen nach ClassPad Protokoll
** Datum:   21.10.2009, v1.2.3-4
** gcc Version: 4.1.3 20080612 (prerelease) (SUSE Linux)
** Lizenz:  GNU GENERAL PUBLIC LICENSE
**
** Copyright (C) 2009,  Johannes Höntsch, j.hoentsch@gmx.de
**
**    GNU-Auszug zur Geltendmachung der Lizenz
**
***********************************************/

#include <stdlib.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "casio.h"

/************
** Funktion: versendet bis zu 4 Zahlen an ClassPad
**           erzeugt NDd-Header und Pruefsumme(PS)
** Eingabe: bis zu 4-stellige Zahl
** Rueckgabe: keine
** Johannes Höntsch, 17.10.2009, v2.6.-18
************/
void cp_send(int wert) //Unterprogramm zum Senden
 {

/* Variablen Definition */
  int16_t c = 0;        //UART-Variable
  uint8_t sc_daten = 0; //Var fuer Switch-Case der Daten und Fuellung
  uint8_t sc_header = 0; //Var fuer Switch-Case des Ndd-Headers

  char checksm_chr[1];  //fuer PS als Zeichen
  char st_1_chr[4];     //fuer einzelne Ziffern als Zeichen,
```

```
37   char st_2_chr[4];     //muessen entsprechend der moeglichen Stellen
38   char st_3_chr[4];     //Platz haben
39   char st_4_chr[4];
40
41   int8_t checksm = 0;   //Pruefsummenvariable
42   int8_t st_1 = 0;      //fuer einzelne dezimale Stellen
43   int8_t st_2 = 0;
44   int8_t st_3 = 0;
45   int8_t st_4 = 0;
46   int8_t st_1_ad = 0;   //fuer in ASCII-Dez gewandelte
47   int8_t st_2_ad = 0;
48   int8_t st_3_ad = 0;
49   int8_t st_4_ad = 0;
50
51
52 /* Temperatur in Ziffern; PS berechnen*/
53
54 /*Pro Ziffer eine Variable vergeben */
55   st_1 = wert / 1000;         //Integerdivision -> 1. Stelle
56   st_2 = (wert % 1000) /100;  //Modulo → letze drei Stellen
57   st_3 = (wert % 100) /10;    //Modulo → letze zwei Stellen
58   st_4 = (wert % 10);         //Modulo enthält letze Stelle
59
60
61 /*in ASCII-Dez "umwandeln"*/
62   st_1_ad = st_1 + 48;
63   st_2_ad = st_2 + 48;
64   st_3_ad = st_3 + 48;
65   st_4_ad = st_4 + 48;
66
67
68 /*0 ausser in Zahl rausnehmen und switch-case-Var für Dateisenden/NDd-
   Header festlegen*/
69   if( (wert >= 0) & (wert <= 9) )         //Zahl einstellig?
70    {st_1_ad = 0x00; st_2_ad = 0x00; st_3_ad = 0x00;
71     sc_daten = 1; sc_header = 1;}         //10er, 100er, 1000er -> 0x00
72
73   if( (wert >= 10) & (wert <= 99) )       //Zahl zweistellig?
74    {st_1_ad = 0x00; st_2_ad = 0x00;
75     sc_daten = 2; sc_header = 1;}         //100er, 1000er -> 0x00
76
77   if( (wert >= 100) & (wert <= 999) )     //Zahl dreistellig?
78    {st_1_ad = 0x00; sc_daten = 3; sc_header = 1;} //1000er ->0x00
79
80   if( (wert >= 1000) & (wert <= 1023) )   //Zahl vierstellig?
81    {sc_daten = 4; sc_header = 1;}         //anderer NDd, andere Fuellung
82
83
84 /*PS berechnen*/
85   checksm = 59 + st_1_ad + st_2_ad + st_3_ad + st_4_ad;
86   checksm = 58 - checksm;
87   checksm = checksm - 0xFFFFFF00; //8-Bit-Register-Ueberlauf entfernen
88
89 /*Ergebnisse als Zeichenketten um per UART auszugeben*/
90   itoa(checksm,checksm_chr,10);     //Pruefsumme
91   itoa(st_1,st_1_chr,10);           //Tausender
92   itoa(st_2,st_2_chr,10);           //Hunderter
93   itoa(st_3,st_3_chr,10);           //Zehner
94   itoa(st_4,st_4_chr,10);           //Einer
95 /* Daten nach ClassPad-Protokoll senden:
96 *
97 * Zwischen Daten und PS steht immer Abstandhalter (0x00),
98 * werden bis zu 3 Stellen gesendet sind es 8 Datenbyte
```

```
 99 * ab 4 Stellen Nutzbyte 12 Datenbyte -> anderer NDd-Header
100 * Datenbit: alle Bytes bei Sendevorgang
101 * Nutzbit: die davon zur Datenuebertragung Nutzbaren
102 */
103
104     uart_putc(0x15); //Handshake beginnen
105
106     do { c = uart_getc(); } //auf Antwort von ClassPad warten
107     while( c != 0x13 );
108
109       uart_puts(":NDd"); //NDd-Header Anfang
110       uart_putc(0x00);
111       uart_putc(0x01);
112       uart_putc(0x00);
113       uart_putc(0x01);
114       uart_putc(0x00);
115
116       switch(sc_header)
117         {
118        case 1:
119            uart_putc(0x06); //Header fuer 1 bis 3 Nutzbit
120            uart_putc(0x00);
121            uart_putc(0x06);
122            uart_putc(0x05);
123            uart_putc(0xFF);
124            uart_putc(0xF8);
125            break;
126
127        case 2:
128            uart_putc(0x0A); //Header fuer 4 Nutzbit
129            uart_putc(0x00);
130            uart_putc(0x0A);
131            uart_putc(0x05);
132            uart_putc(0xFF);
133            uart_putc(0xF0);
134            break;
135       }
136
137     do { c = uart_getc(); } //wenn Bestaetigung
138     while ( c != 0x06 );
139     c = 0; //UART-Var löschen, sonst wird an Daten letzte Eingabe gehangen
140
141       uart_putc(0x3A); //Daten senden; Datenanfang
142       uart_putc(0x00);
143       uart_putc(0x01);
144
145     switch(sc_daten)
146      {
147       case 1:                 //bei einer Stelle
148         uart_puts(st_4_chr);  //Einer
149         uart_putc(0x00);      //Fuellung
150         uart_putc(0x00);
151         uart_putc(0x00);
152         break;
153
154       case 2:                 //bei zwei Stellen
155         uart_puts(st_3_chr);  //Zehner
156         uart_puts(st_4_chr);  //Einer
157         uart_putc(0x00);      //Fuellung
158         uart_putc(0x00);
159         break;
160
161       case 3:                 //bei drei Stellen
```

```
162         uart_puts(st_2_chr);  //Hunderter
163         uart_puts(st_3_chr);  //Zehner
164         uart_puts(st_4_chr);  //Einer
165         uart_putc(0x00);      //Fuellung
166         break;
167 
168       case 4:                 //bei vier Stellen
169         uart_puts(st_1_chr);  //Tausender
170         uart_puts(st_2_chr);  //Hunderter
171         uart_puts(st_3_chr);  //Zehner
172         uart_puts(st_4_chr);  //Einer
173         uart_putc(0x00);      //Fuellung
174         uart_putc(0x00);
175         uart_putc(0x00);
176         uart_putc(0x00);
177         break;
178 
179      default:
180         uart_putc(0x21); //Fehlercode senden
181         break;
182      }
183 
184       uart_putc(checksm); //Pruefsumme senden
185 
186      do { c = uart_getc(); } //wenn Bestaetigung → Programmende
187      while ( c != 0x06 );
188  }
189 
190 
191 /************
192 ** Funktion: Empfaengt ein ClassPad-String bis zu 3 Zahlen via UART
193 **
194 ** Eingabe: keine
195 ** Rueckgabe: Zahlen (Daten) als Integer
196 ** Johannes Hoentsch, 21.10.2009, v1.5.5-53
197 ************/
198 
199 int cp_get(void)
200  {
201   uint8_t i = 0;         //Zaehlvariable
202   uint8_t st_1 = 0;      //Variable fuer erste Stelle im uart_getc()
203   uint8_t st_2 = 0;      //Variable fuer zweite Stelle im uart_getc()
204   uint8_t st_3 = 0;      //Variable fuer dritte Stelle im uart_getc()
205   uint16_t checksm = 0; //Variable fuer Pruefsumme
206   uint16_t wert = 0;     //Variable fuer Endergebnis
207   int16_t c = 0;         //UART-Variable
208 
209   do { c = uart_getc();} //auf Kommunikation warten
210   while ( c != 0x15 );
211 
212   uart_putc(0x13);       //NDd-Header erfragen
213 
214   do { c = uart_getc();} //warte auf erstes Zeichen vom NDd-Header
215   while ( c != 0x3A );
216   for( i = 0; i <= 13; i++ ) // verwerfe NDd-Header
217    {
218     while ((c = uart_getc()) & UART_NO_DATA) //UART lesen, keine Daten?
219      {;}  //warte auf Daten
220     st_1 = c; //muss aus Empfangs-Puffer, wird dann wieder ueberschrieben
221    }
222 
223   uart_putc(0x06);
224   c = 0; //UART-Variable zuruecksetzen, sonst ist NDd-Header im Buffer
```

```
225   st_1 = 0; //Stelle 1 zuruecksetzen
226
227 /*[Daten] : 0x3A 0x01 0x00 0xaa 0xbb 0xcc 0x00 0xPS -> aa, bb, cc, PS
    extrahieren*/
228   do { c = uart_getc();} //warte auf erstes Zeichen vom Datenstrom
229   while ( c != 0x3A );
230
231   for( i = 0; i != 2; i++ ) // 0x00 und 0x01 verwerfen (0x3A schon weg)
232    {
233     while ((c = uart_getc()) & UART_NO_DATA) //UART lesen, keine Daten?
234     {;}  //warte auf Daten
235     st_1 = c; //muss aus UART-Empfangs-Puffer
236    }
237
238 /*erstes Zeichen nach Stelle 1 schreiben*/
239   st_1 = 0; //Ruecksetzen
240   while ((c = uart_getc()) & UART_NO_DATA) //UART lesen, keine Daten?
241     {;}  //warte auf Daten
242     st_1 = c; //sonst schreibe Zeichen nach Stelle 1
243
244 /*zweites Zeichen nach Stelle 2 schreiben*/
245   while ((c = uart_getc()) & UART_NO_DATA) //UART lesen, keine Daten?
246     {;}  //warte auf Daten
247     st_2 = c; //sonst schreibe Zeichen nach Stelle 2
248
249 /*drittes Zeichen nach Stelle 3 schreiben*/
250   while ((c = uart_getc()) & UART_NO_DATA) //UART lesen, keine Daten?
251     {;}  //warte auf Daten
252     st_3 = c; //sonst schreibe Zeichen nach Stelle 3
253
254   for( i = 0; i != 2; i++ ) //Abstandhalter verwerfen, PS speichern
255    {
256     while ((c = uart_getc()) & UART_NO_DATA) //UART lesen, keine Daten?
257     {;}  //warte auf Daten
258     checksm = c; //0x00 (verworfen), PS -> bleibt in Variable
259    }
260
261   uart_putc(0x06); //Bestätige Empfang
262
263 /*Ergebnis "wert" bilden*/
264   if(st_2 == 0x00) //Stelle 2 ohne Wert
265    {wert = st_1 - 48;} //Wert von Stelle 1, minus ASCII '0'
266   else
267    {
268     if (st_3 == 0x00) //Stelle 3 ohne Wert
269      {wert = 10 * (st_1 - 48) + (st_2 - 48);} //jeweils minus ASCII '0'
270     else //ueberall Werte
271      {wert = 100 * (st_1 - 48) + 10 * (st_2 - 48) + (st_3 - 48);}
272    }
273
274   _delay_ms(100); //kurz warten
275   return wert; //Zahl an Hauptprogramm zurueckgeben
276  }
```

QC06 – Hauptmessprogramm

```
1  /*Hauptprogramm
2  ** - holt Messzeit
3  ** - holt Messintervall -> ermittelt Messtyp
4  ** - ermittelt ADC-Wert
5  ** - uebergibt an Unterprogramm, das nach CP-Protokoll versendet
6  **
7  **   Johannes Hoentsch, v2.1.3-18 21.10.2009
8  */
9
10 #include <stdlib.h>
11 #include <avr/io.h>
12 #include <avr/interrupt.h>
13 #include <avr/signal.h>
14 #include <avr/pgmspace.h>
15 #include <util/delay.h>
16
17 #include "casio.h"
18
19 #ifndef F_CPU
20 #define F_CPU 8000000UL
21 #endif
22
23
24 int main(void)
25  {
26
27 /*Variablendeklaration*/
28   uint16_t adc = 0;     //Ergebnisvariable
29   uint8_t i = 0;        //Zaelvariable
30   uint8_t messung = 0;  //Zaehlvar fuer Anzahl der Messungen
31   uint16_t intervall = 0; //Variable fuer Messintervall
32   uint16_t zeit = 0;    //Variable fuer Messzeit
33   uint8_t messtyp = 0;  //0 fuer sofort, 1 fuer Diagramm
34
35   DDRB = 0xFF; //Port mit LEDs zur Messtyp-Anzeige als Ausgang deklarieren
36
37 /* A/D-Register setzen */
38   ADMUX = 0; //von ADC0 lesen
39   ADMUX |= (1<<REFS0); // A/D-Referenzbits auf AVCC setzen
40   ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1); // A/D-Wandler aktiv
41
42 /* UART-Init */
43   uart_init( UART_BAUD_SELECT_DOUBLE_SPEED(UART_BAUD_RATE,F_CPU) );
44   sei();
45
46 /* 1 Messung zum Anlaufen */
47   ADCSRA |= (1<<ADSC); // A/D-Wandlung starten
48   while (ADCSRA & (1<<ADSC)) //solange Konvertierung laeuft
49    { ;} //mache nichts
50
51   adc = ADCW; //Wert nach "adc" schreiben
52
53 for(;;) //Hauptmessschleife – wird nicht verlassen
54  {
55   PORTB = 0x00; //Messtyp-LED (wieder) aus
56
57   intervall = 0;  //Variablen zuruecksetzen
58   zeit = 0;
59   messung = 0;
60   adc = 0;
```

```
61
62 /*Messintervall und -zeit holen, Messtyp festlegen*/
63   intervall = cp_get();
64   zeit = cp_get();
65
66   if(intervall == 0) //wird fuer Sofortmessung gesendet
67    {messtyp = 0;} //entsprechend einstellen
68   else
69    {messtyp = 1;}
70
71 /*Messschleifen*/
72   switch(messtyp)
73    {
74     case 0:              //fuer eine Sofortmessung
75       PORTB = 0x01;      //LED Sofortmessung an
76       adc = 0;           //Ergebnis zuruecksetzen
77       for(i=0; i<16; i++) //16x messen
78        {
79         ADCSRA |= (1<<ADSC);         // A/D-Wandlung starten
80         while (ADCSRA & (1<<ADSC))  //Solange Konvertierung laeuft
81          { ;}                       //mache nichts
82         adc = adc + ADCW;           //16 Messungen aufsummieren
83        }
84       i=0;               //Zaehlvariable reset
85       adc = (adc>>4);    //Mittelwert aus 16 Messungen bilden
86       cp_send(adc);      //Spannung per CP-Sende-Routine ausgeben
87     break;
88
89     case 1:              //fuer Diagrammessung
90     PORTB = 0x02;        //LED Diagrammmessung an
91       while(messung < zeit)
92        {
93         adc = 0;  //Ergebnis zuruecksetzen
94
95         for(i=0; i<16; i++)   //16x messen
96          {
97           ADCSRA |= (1<<ADSC);            // A/D-Wandlung starten
98           while (ADCSRA & (1<<ADSC))      //Solange Konvertierung laeuft
99            { ;}                           //mache nichts
100          adc = adc + ADCW;         //16 Messungen aufsummieren
101         }
102         i=0;            //Zaehlvariable reset
103
104         adc = (adc>>4); //Mittelwert aus 16 Messungen bilden
105         cp_send(adc);   //Spannung per CP-Sende-Routine ausgeben
106         messung += intervall;       // erhoehe Messdurchlauf
107
108 /* ein Intervall abwarten; Schleife, da _delay_ms() max 6,5s warten kann */
109       while(i<intervall) //so oft je 1 s abwarten, bis Intervall erreicht
110        { _delay_ms(1000);
111          i++;
112        }
113       i=0; //Zaehlvariable reset
114       }
115    break;
116    }
117 }
118 }
```

QC07 – einzelne Unterprogramme des Temperaturprogramms (alphapetisch)

Name: alldel

Funktion: löscht alle Variablen und Einstellungen des Programms

```
 1 Message "Achtung! Alle Einstellungen des Temperaturprogramms werden
 2 gelöscht!","Warnung"
 3 ClrText
 4
 5 Locate 1,1, "lösche Variablen"
 6 Locate 1,10, "stelle System wieder her"
 7 vardel()
 8 DelVar eing
 9 DelVar stdda
10 DelVar polyok
11
12 Locate 1,20, "Lösche Standarddefinitionen"
13 DelVar zeit,tdif
14 DelVar mindia,maxdia
15 Unlock std
16 DelVar std
17
18 Locate 1,30, "Lösche Berechnungspolynom"
19 DelVar polya,polyb,polyc,polyd,polye,polyf
20 DelVar funktion
21
22 Locate 1,45, "Fertig"
23 Locate 1,60, "zum Beenden 'Menu' klicken"
24
25 Stop              //Beendet Programm
```

Name: atoi

Funktion: wandelt einen String in eine Zahl um

```
 1 StrLen adc, l           //bestimme Laenge von adc
 2 1 => i                  //Stellenvariable
 3 0 => z                  //Ergebnis
 4 While i<l+1             //Solange nicht alle Stellen durchgegangen
 5  ChrToNum adc, b, i     //Wandle Zeichen an Position i aus adc in b um
 6  10×z+(b-48) => z       //setze Zahl zusammen
 7  i+1 => i               //naechse Position
 8 WhileEnd
 9
10 DelVar l, adc, b, i
11 Return z          //gebe Ergebnis zurück
```

Name: ausgd

Funktion: zeichnet ein Diagramm der gemessenen Temperaturen und gibt diese aus

```
1 Message "Messung starten?"  //wenn bestaetigt wird, startet Messung
2
3 SetDecimal
4 ViewWindow 0,(zeit-tdif),1,mindia,maxdia,1//Diagrammfenster einstellen
5 ClrText
```

```
 6
 7 OpenComPort38k
 8 SendVar38k tdifsend          //Parameter senden
 9 SendVar38k zeitsend
10
11 /*Messung f. t=0 und erste Temperatur -> nur so line(x1,y1;x2,y2) moegl.*/
12 GetVar38k adc                //1. AD-Wert holen
13 atoi()                       //Wert wandeln
14 polya×z^5+polyb×z^4+polyc×z^3+polyd×z^2+polye×z+polyf => y_alt
15 {y_alt} => list2             //1. Wert in y-Liste eintragen
16 Print y_alt                  //Wert anzeigen
17 x + interv => x              //naester Messpunkt
18
19 While x<zeit           //fuer alle restlichen Messungen folgendes:
20  GetVar38k adc         //hole aktuellen AD-Wert
21  atoi()                //wandle in Zahl
22  polya×z^5+polyb×z^4+polyc×z^3+polyd×z^2+polye×z+polyf => y
23  Print y               //gebe aktuellen Wert aus
24  Line x-tdif,y_alt,x,y //Linie zu letztem Punkt ziehen
25  x+tdif => x           //Messdurchlauf um Intervall erhoehen
26  CopyVar y, y_alt      //kopiere vorherigen Wert
27  {y} => tempa          //speichere Wert in Liste der aktuellen Temperatur
28  augment(list2,tempa) => list2   //zu 'Temperaturliste' hinzufuegen
29 WhileEnd
30
31 CloseComPort38k
32
33 {1E-10} => list1             //1. Wert für Liste 'Zeit' fast 0
34 seq(x,x,tdif,zeit,tdif) => listzeit //Zeit-Liste erstellen (s. QC10)
35 augment(list1, listzeit) => list1
36
37 DelVar tempa
38
39 regres()              //Unterprogramm zur Auswertung oeffnen
```

Name: diagra

Funktion: überprüft Statusvaribale und reagiert entsprechend

```
1 If std = 1              //Wenn Standard verwendet werde soll
2  Then:ausgd()           //gehe direkt zu Diagrammausgabe
3 ElseIf std = 2          //oder wenn Standard nciht verwendet werden soll
4  Then:wert():ausgd()    //Einstellungen vornemen, dann Diagrammausgabe
5 Else:Message "Falsches Argument für 'Standard'":temp1() //Fehlerausgabe
6 IfEnd
```

Name: einst

Funktion: Anzeige und Auswahl zum Ändern der Einstellungen

```
1 ClrText
2 SetNormal 1                  //Anzeige von Kommastellen
3
4 Locate 5,1, "1 - Standard aktivieren"
```

```
 5 Locate 5,11, "2 - Standard deaktivieren"
 6 Locate 5,21, "3 - Standard ändern"
 7 Locate 5,31, "4 - Polynom ändern"
 8
 9 Locate 3,44, "Aktueller Standard:"
10 Locate 12,55, "untere:"
11 Locate 54,55, mindia
12 Locate 85,55, "obere:"
13 Locate 122,55, maxdia
14 Locate 12,65, "Messintervall:"
15 Locate 90,65, tdif
16 Locate 12,75, "Messdauer:"
17 Locate 74,75, zeit
18
19 Unlock std              //Variable entsperren
20 Input std, "Auswahl:"   //Variable eingeben
21 Lock std                //Variable wieder sperren
22
23 Switch std
24  Case 1:                      //Standard aktivieren
25   GetType mindia, minstat
26    If minstat = "STR"         //Standard vorhanden?
27     Then: Message "Bitte Standard definieren." : wert()
28     Else: temp1()             //zurueck zu Hauptauswahl
29    IfEnd
30  Case 2: temp1()              //sofort zurueck zu Hauptauswahl
31  Case 3: wert():einst()       //Werte aendern, dann zurueck zu Einstellungen
32  Case 4: polyaend()           //Polynom aendern (geht automatisch zurueck)
33 SwitchEnd
```

Name: e_interv

Funktion: Einstellungsdialog des Messintervalls

```
 1 e_skizze()              //Schema anzeigen
 2
 3 Lbl tdif_neu            //Einsprung bei Fehleingabe
 4 Input tdif, "Messintervall, [2s;999s]"
 5 If tdif < 2
 6  Then: Message "Wertebereich unterschritten"
 7        Goto tdif_neu
 8  ElseIf tdif > 999
 9   Then: Message "Wertebereich überschritten"
10         Goto tdif_neu
11 IfEnd
12
13 ExpToStr tdif,tdifsend  //zum spaeteren Senden wandeln
```

Name: e_obere

Funktion: Einstellungsdialog zum Ändern der oberen Grenze in der Diagrammdarstellung

```
 1 e_skizze()        //Schema anzeigen
 2
 3 Locate 4,18, "-|- obere"
 4 Lbl max_neu       //Einsprung bei Fehleingabe
 5 Input maxdia, "obere Temperatur, [-29;+130]"
```

```
 6
 7 If maxdia < -29
 8  Then: Message "Wert zu klein"
 9        goto max_neu
10 ElseIf maxdia > 130
11  Then: Message "Wert zu groß"
12        goto max_neu
13 IfEnd
14
15 If mindia = maxdia
16  Then: Message "Achtung, Werte für obere und untere Grenze sind gleich!"
17 ElseIf mindia > maxdia
18  Then: Message "Achtung, der Wert für die obere ist kleiner als der Wert
19                 für die untere Grenze!"
20        e_obere()  //nochmal alles aendern
21        e_untere()
22 IfEnd
23
24 Locate 20,18, " " //in Skizze eigegebenen Wert uebernehmen
25 Locate 25,18, maxdia
```

Name: e_skizze

Funktion: Anzeige eines beschrifteten Schemas für das Diagramm

```
 1 ClrText
 2
 3 Locate 10,2, "^ υ [℃]"
 4 Locate 10,9, "|"
 5 Locate 4,18, "-|- obere"
 6 Locate 10,27, "|"
 7 Locate 10,36, "|"
 8 Locate 4,45, "-|- untere"
 9 Locate 10,45, "|"
10 Locate 12,54, "------------------->"
11 Locate 12,61, "  'Messdauer'     t[s]"
12
13 Locate 20,45, " "           //Werte anzeigen
14 Locate 25,45, mindia
15
16 Locate 20,18, " "
17 Locate 25,18, maxdia
```

Name: e_untere

Funktion: Einstellungsdialog der unteren Grenze in der Diagrammdarstellung

```
 1 e_skizze()         //Schema anzeigen
 2
 3 Locate 4,18, "-|- obere"
 4 Locate 4,45, "-|- untere"
 5 Lbl min_neu        //Einsprungmarke bei Fehleingabe
 6 Input mindia, "untere Temperatur, [-30;+129]"
 7 If mindia < -30
```

```
 8  Then: Message "Wert zu klein"
 9        goto min_neu
10  ElseIf mindia > 129
11   Then: Message "Wert zu groß"
12         goto min_neu
13 IfEnd
14
15 Locate 20,45, " "   //neuen Wert in Schema eintragen
16 Locate 25,45, mindia
```

Name: e_zeit

Funktion: Einstellungsdialog der Messzeit

```
 1 e_skizze()        //Schema anzeigen
 2
 3 Lbl zeit_neu      //Einsprung bei Fehleingabe
 4 Input zeit, "Messdauer [s], [Messintervall;999]"
 5 If zeit < tdif
 6  Then: Message "Messdauer muss ≥ Messintervall sein."
 7        Goto zeit_neu
 8  ElseIf tdif > 999
 9   Then: Message "Wertebereich überschritten"
10         Goto zeit_neu
11 IfEnd
12
13 ExpToStr zeit,zeitsend //fuer spaeteres Senden umwandeln
```

Name: init

Funktion: initialisiert Programm, führt Statusabfragen und ggf. Ausgaben durch

```
 1 GetType std, stdda      //Variablentyp abfragen
 2 If stdda = "NONE"       //Variable nicht vorhanden?
 3  Then: 0 => std : Lock std    //lege an und sperre sie
 4        Message "Erster Programmstart oder vorheriger Reset: kein Standard
                   und Polynom definiert. Es wird jetzt ein Fenster geöffnet,
                   um dies zu ändern.", "Willkommen"
 5        "-" => mindia    //Diagrammwerte als n.df. einstellen
 6        "-" => maxdia
 7        "-" => tdif
 8        "-" => zeit
 9        wert()           //Unterprogramm zum Werte einstellen
10        polyaend()       //Unterprogramm zum Polynom vergeben
11 IfEnd
12
13 Local temp,adc,eing     //temporaere Variablen anlegen
14 0 => x
15 SetNormal 1
16
17 Return
```

Name: polyaend

Funktion: Eingabe eines neuen Berechrechnungspolynoms

```
ClrText
Locate 2,1, "Eingabe des Berechnungs-"
Locate 2,11,"polynoms der Form"
Locate 5,25, "a*x⁵+b*x⁴+c*x³+d*x²+e*x+f"
Locate 2,50, "i Für nicht verwendete"
Locate 2,61,"  Koeffizienten `0' eigeben."
Input polya, "a"
Input polyb, "b"
Input polyc, "c"
Input polyd, "d"
Input polye, "e"
Input polyf, "f"

polyprob()
If polyok = 0
 Then: Message "Kein gültiges Polynom definiert!","Fehler!"
       polyaend()
IfEnd

einst()          //zurueck zu den Einstellungen
```

Name: polyprob

Funktion: überprüft eingegebenes Polynom, setzt dann Statusvariable

```
If (polya = 0) and (polyb = 0) and (polyc = 0) and (polyd = 0) and
(polye = 0) and (polyf = 0)  //gueltiges Polynom vorhanden?
 Then: 0 => polyok //Polynom ist ungueltig
 Else: 1 => polyok //Polynom ist gueltig
IfEnd
```

Name: regres

Funktion: Regressionsmöglichkeiten anzeigen und nach Auswahl durchführen

```
Lbl start
ClrText
Print "Regressionsformeltypen:"
Print "1 - y=ax+b"
Print "2 - y=ax²+bx+c"
Print "3 - y=ax³+bx²+cx+d"
Print "4 - y=ax⁴+bx³+cx²+dx+e"
Print "5 - y=a*b^(x)"
Print "6 - y=a+b*ln(x)"
Print "7 - y=a*e^(b*x)"
Print "8 – y=a*x^(b)"

Input funktion,"Bitte Zahl des Funktionstyps eingeben"

/*Je nach gewaehlter Formel Regression durchfuehren, Ergebnis anzeigen,
Funktion zeichnen*/
Switch funktion
 Case 1: LinearReg list1,list2:DispStat:DrawStat:Break
 Case 2: QuadReg list1,list2:DispStat:DrawStat:Break
```

```
 Case 3: CubicReg list1,list2:DispStat:DrawStat:Break
 Case 4: QuartReg list1,list2:DispStat:DrawStat:Break
 Case 5: abExpReg list1,list2:DispStat:DrawStat:Break
 Case 6: LogReg list1,list2:DispStat:DrawStat:Break
 Case 7: ExpReg list1,list2:DispStat:DrawStat:Break
 Default: Message "Auswahlfehler!":Goto start:Break
SwitchEnd

vardel() //loescht alle genutzten Variablen
Stop  //Programmende
```

Name: temp1

Funktion: Hauptauswahl des Temperaturprogramms, zur Navigation in Unterprogramme

```
Init()

ClrText
Locate 10,1, "Temperaturmessprogramm"
Locate 5,15, "1 - eine Sofortmessung"
Locate 5,25, "2 - Diagramm"
Locate 5,35, "3 – Einstellungen"
Locate 5,45, "4 - Variablen löschen"
Locate 5,55, "5 - Alles zurücksetzen"

If std = 1      //Informationen zur 'Standard'-Einstellung auslesen
 Then: Locate 10,80, "Standard aktiv: Ja"
ElseIf std = 2
 Then: Locate 10,80, "Standard aktiv: Nein"
ElseIf std = 0
 Then: Locate 10,80, "Standard aktiv: n. df."
Else: Locate 10,80, "Standard aktiv: Fehler!"
IfEnd

polyprob()
If polyok = 0
Then: Locate 10,80, "Polynom gültig: Nein"
ElseIf polyok = 1
Then: Locate 10,80, "Polynom gültig: Ja"
Else: Locate 10,80, "Polynom gültig: Fehler!"
IfEnd

Input eing, "Auswahl:"

If eing = 1
 Then: wand1()    //Unterprogramm fuer Sofortmessung
ElseIf eing = 2
 Then: diagr()    //Unterprogramm fuer Diagrammmessung
ElseIf eing = 3
 Then: einst()    //Unterprogramm fuer Einstellungen
ElseIf eing = 4
 Then: goto vardel //Unterprogramm zum Variablen loeschen
ElseIf eing = 5
 Then: alldel()   //Unterprogramm fuer Reset
Else: Message "Auswahlfehler" : temp1()
IfEnd

Lbl vardel //vardel durchfuehren und Ausgabe zeigen
ClrText
vardel()
Locate 1,25, "Fertig"
```

```
47 Locate 1,40, "zum Beenden 'Menu' klicken"
48 Stop          //Programm beenden
```

Name: vardel

Funktion: löscht alle Nutzvariablen, keine Einstellungen

```
 1 DelVar eing,funktion
 2 DelVar schl,stdda
 3 DelVar einstvar,tdif2
 4 DelVar zeit2,minstat
 5 DelVar zeit,tdif
 6 DelVar listzeit, y_alt
 7 Clear_a_z
 8
 9 SetNormal 1
10 ViewWindow -10,10,1,-7,7,1           //Diagrammfenster auf Standard
```

Name: wand1

Funktion: führt eine Wandlung durch und gibt das Ergebnis aus

```
 1 SetDecimal
 2 ClrText
 3
 4 OpenComPort38k
 5 0 => tdif2              //Steuercode fuer Sofortmessung
 6 ExpToStr tdif2,tdif2    //Wandeln zum Senden
 7 SendVar38k tdif2
 8
 9 2 => zeit2      //ist beliebig, da bei Sofortmessung nicht ausgewertet wird
10 ExpToStr zeit2,zeit2
11 SendVar38k zeit2
12
13 SetFix 1          //eine Nackommastelle
14 GetVar38k adc     //Wert holen
15 atoi()
16 polya×z^5+polyb×z^4+polyc×z^3+polyd×z^2+polye×z+polyf => c
17
18 Locate 5,10, "Temperatur:"
19 Locate 73,10, c
20 Locate 110,10, "°C"
21 CloseComPort38k
22
23 Message "Messung abgeschlossen. Zurück zum Hauptprogramm?"
24
25 temp1()          //Hauptauswahl wieder aufrufen
```

Name: wert

Funktion: fragt zu ändernde Einstellung ab und reagiert entsprechend

```
 1 ClrText
 2
 3 Print "1 - untere Grenze"
 4 Print "2 - obere Grenze"
 5 Print "3 - Messintervall"
 6 Print "4 - Messdauer"
 7 Print "5 - Alle"
 8 Print " "
 9 Print "i Bei mehreren Zahlen diese"
10 Print "   hintereinander eingeben."
11 InputStr e_aend, "Welche Einstellung(en) wollen Sie ändern?"
12
13 ClrText
14
15 /*Wenn Zahl eingegeben wurde, wird entsprechende Stelle in Variable
   vermerkt (sonst '0')*/
16 StrSrc e_aend, "1", aeun
17 StrSrc e_aend, "2", aeob
18 StrSrc e_aend, "3", aein
19 StrSrc e_aend, "4", aeze
20 StrSrc e_aend, "5", aeal
21
22 If e_aend > 0       //sollen alle geaendert werden?
23  Then: e_untere():e_obere():e_interv():e_zeit()
24        Goto ende    //Rest ueberspringen
25 IfEnd
26
27 If aeun > 0 //soll untere geaendert werden?
28  Then: e_untere()
29 IfEnd
30
31 If aeob > 0 //soll obere geaendert werden?
32  Then: e_obere()
33 IfEnd
34
35 If aein > 0 //soll Intervall geaendert werden?
36  Then: e_interv()
37 IfEnd
38
39 If aeze > 0 //soll Dauer geaendert werden?
40  Then: e_zeit()
41 IfEnd
42
43 If (aeun = 0)  and (aeob = 0)  and (aein = 0)  and (aeze = 0)  and
44    (aeal = 0) //Eingabefehler (keine oder > 5)
45  Then: Message "Keine oder falsche Eingabe!"
46        wert()          //wieder zum Anfang
47 IfEnd
48
49 Lbl ende    //Einsprung fuer 'alle aendern'
50 DelVar aeun,aeob,aein,aeze,e_aend,aeal
```

QC08 – Beispiel für Abfrage als Polling (ClassPad-Programm)

```
1 getkey eingabe          //schreibe gedrueckte Taste nach eingabe
2 While eingabe != 48     //solange bis Taste '0' (=48) gedrueckt wurde
3  getkey eingabe         //schreibe gedrueckte Taste nach eingabe
4 WhileEnd
5
6 Print "Taste '0' wurde gedrückt"
```

QC09 – Beispiel für Augmentierung (ClassPad-Programm)

```
1 ClrText                       //Bildschirm löschen
2 1 => x
3 {0} => list1                  //Starte Liste1 mit eintrag '0'
4 While x<10                    //bis x=10:
5  x+1 => x                     //erhöhe x
6  {x} => list2                 //schreibe erhoehtes x in Liste2
7  augment(list1,list2) => list1 //Liste1 := Liste1 und am Ende Liste2
8 WhileEnd                      //Ende der Schleife
9 Print list1                   //Testausgabe der erzeugten Liste
```

```
Ausgabe: {0,2,3,4,5,6,7,8,9,10}
```

QC10 – Beispiel zur Listenbildung mittels *seq* (ClassPad-Programm)

```
1 ClrText
2 Input interv
3 Input zeit
4 {1E-10} => list1                          //erster Eintrag muss 'fast 0' sein
5 seq(x,x,interv, zeit, interv) =>listzeit //zaehle x von invterv immer
                                            interv-mal hoch, bis zeit erreicht
6 augment(list1, listzeit) => list1
7 Print list1
```

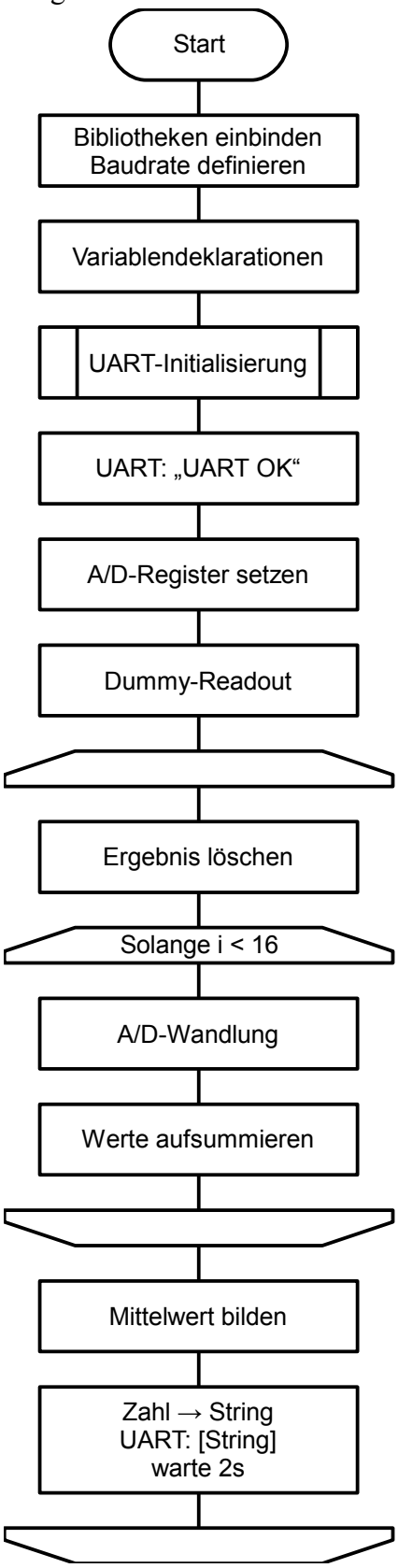
```
Eingabe: interv = 2, zeit = 20
Ausgabe: {1E-10,2,4,6,8,10,12,14,16,18,20}
```

Die vorgenommenen Einrückungen sind nicht in den originalen Programmen enthalten, sie dienen hier nur zur Verbesserung der Übersichtlichkeit.

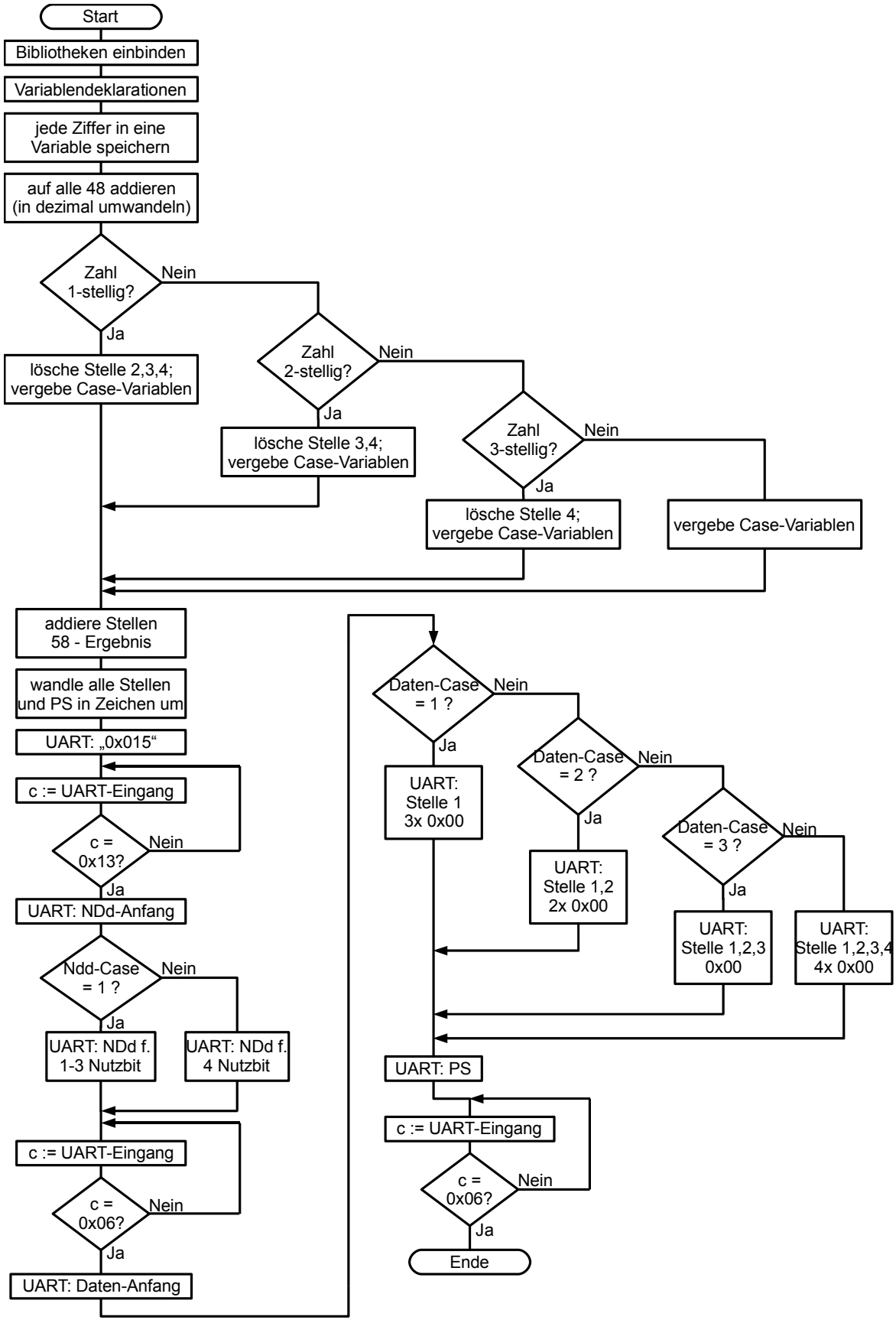
Die Kommentierung erfolgt nach C-Norm. Sie ist ebenfalls (aus Platzgründen) nicht Bestandteil der laufenden Routinen.

XII. 4 Programmablaufpläne

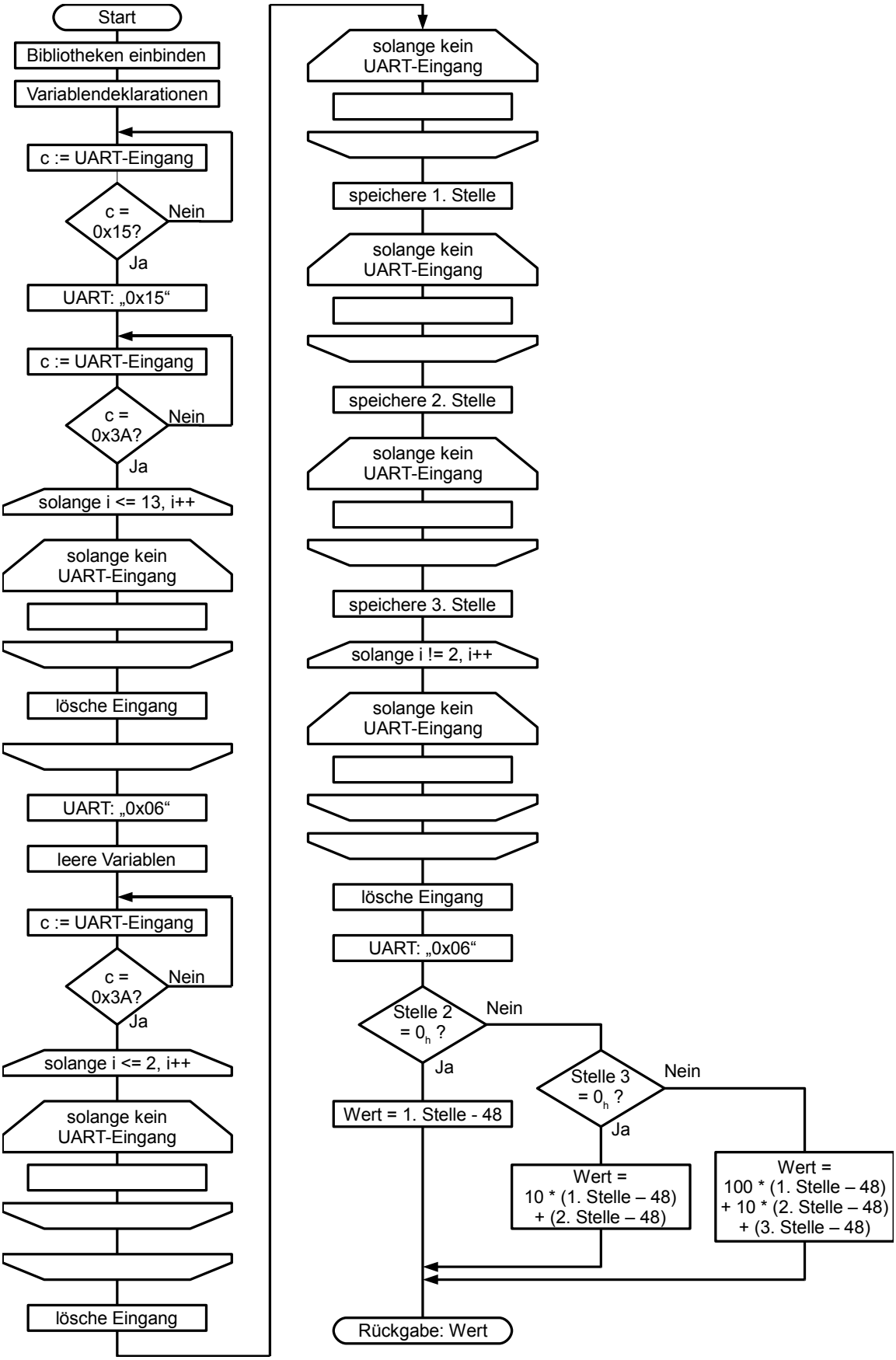
PAP01 – UART-Ausgabe des vorher ermittelten ADCW



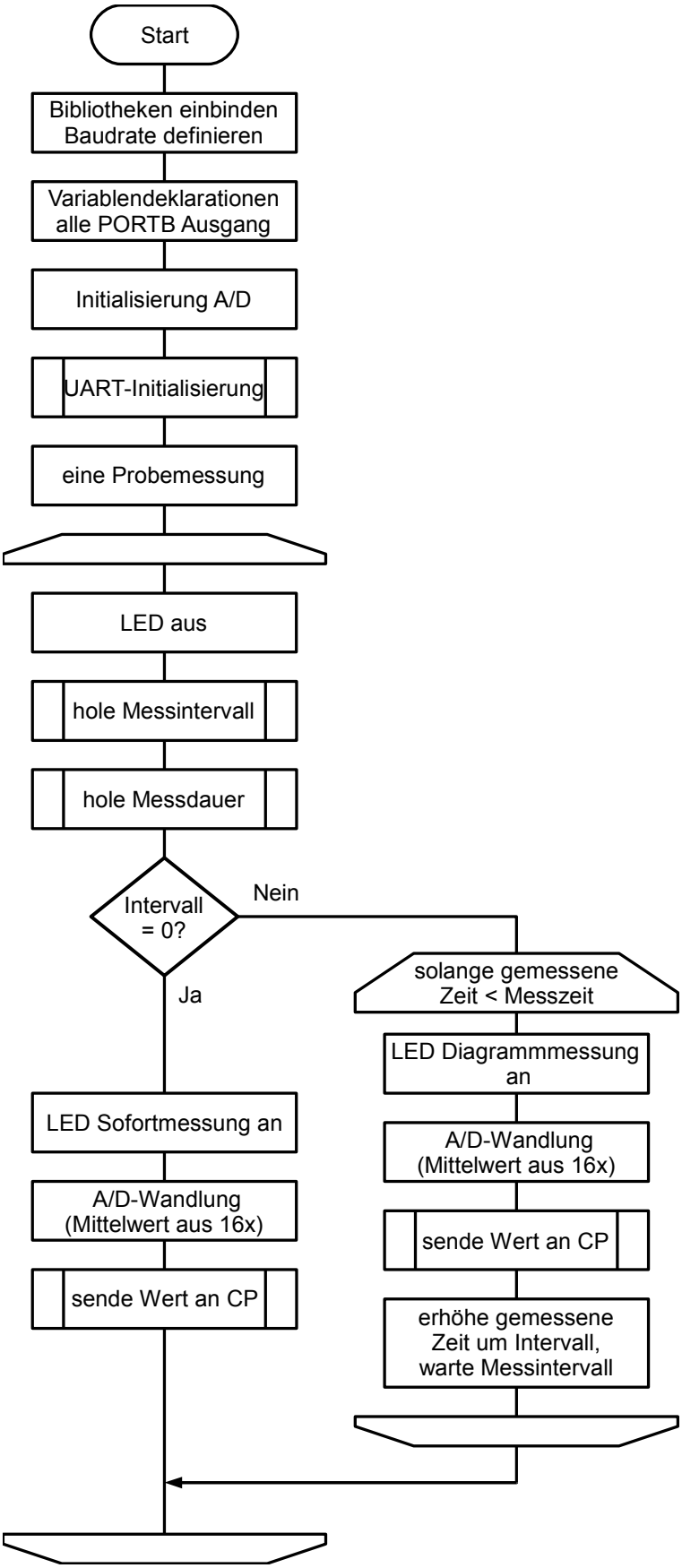
PAP02 a) – Ablauf der casio.c (*cp_send()*)



PAP02 b) – Ablauf der casio.c (*cp_get()*)

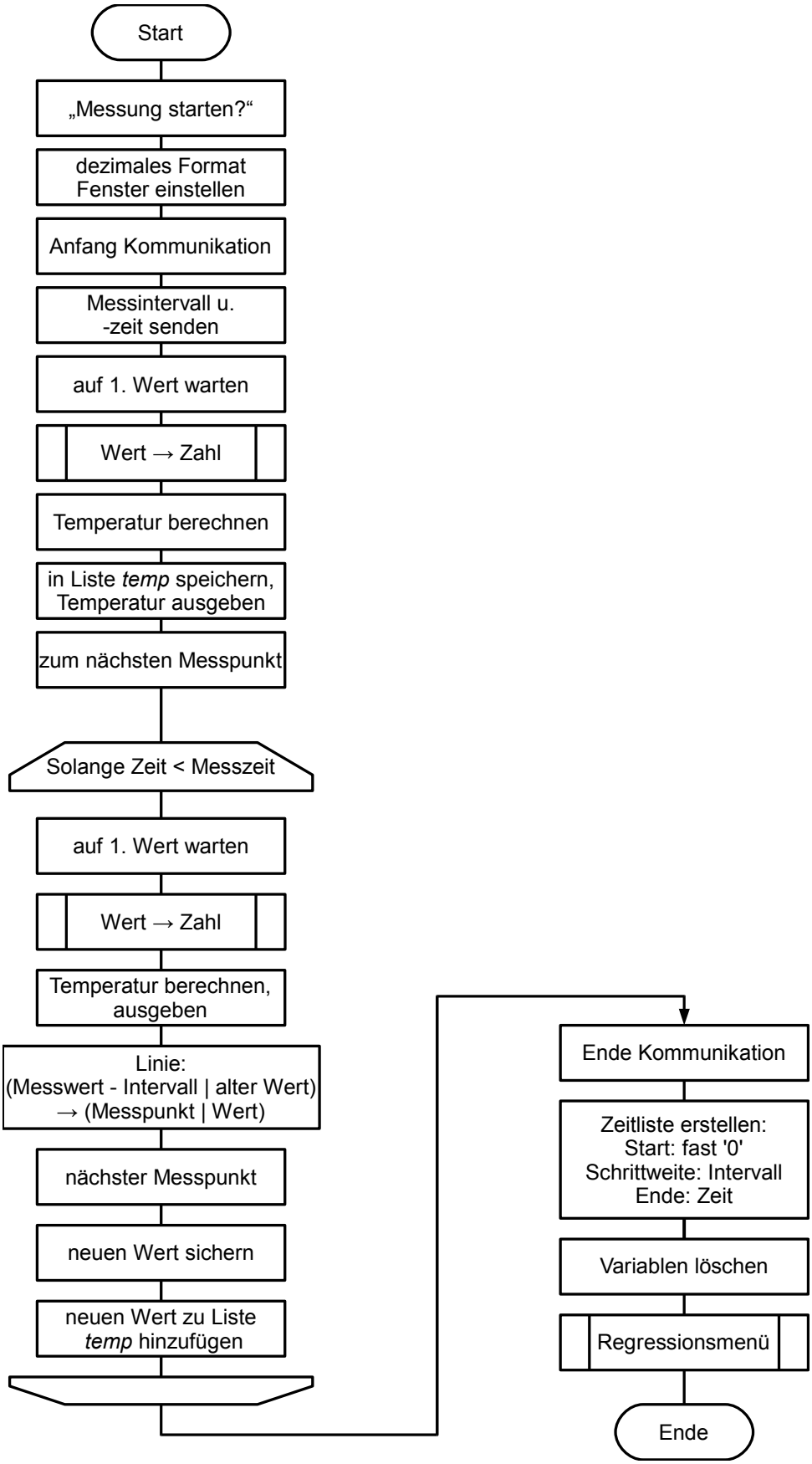


PAP03 – Hauptprogramm des Messcontrollers

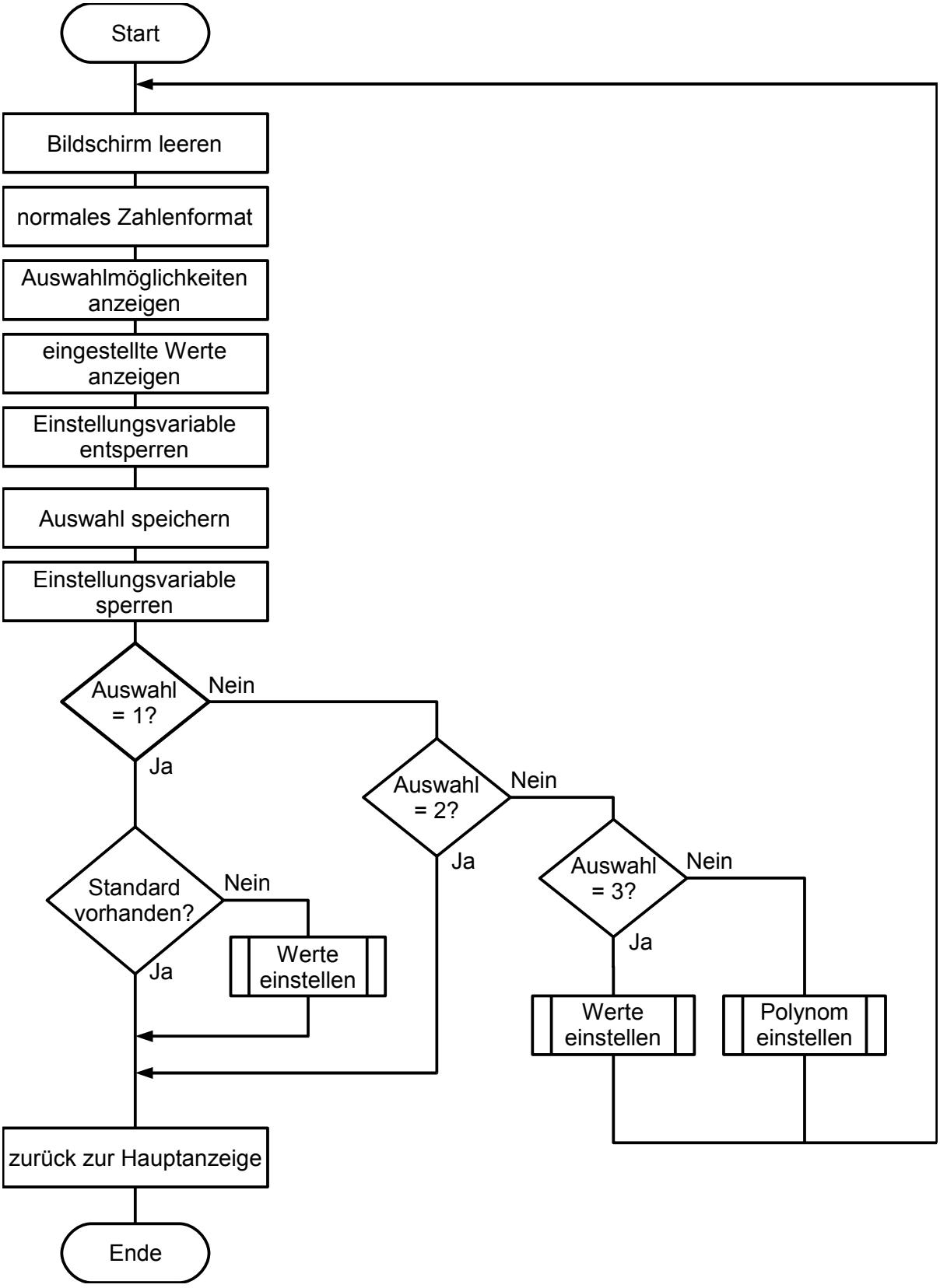


PAP04 – komplexe ClassPad-Programme

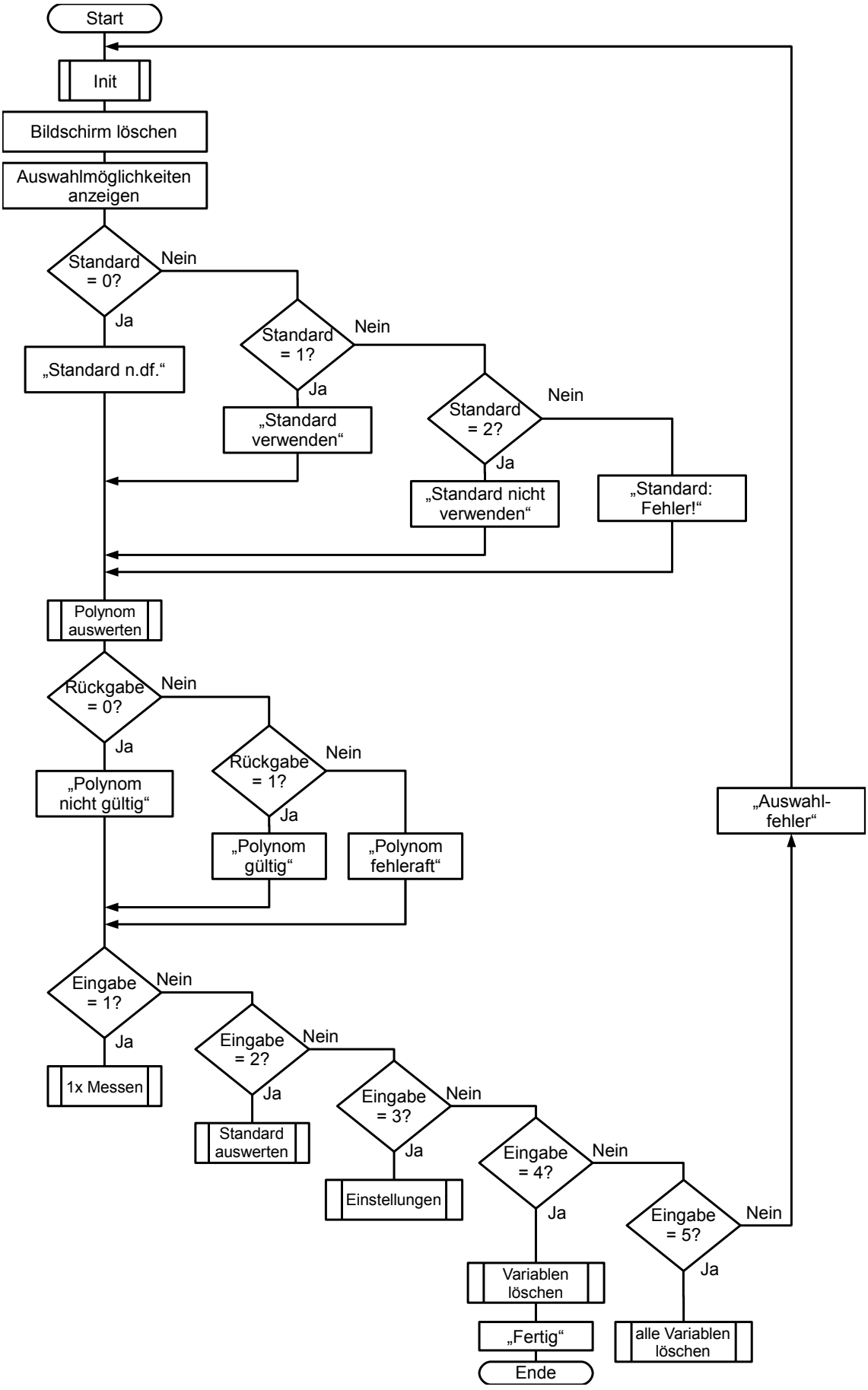
Name: ausgd (Quellcode S.80 f.)



Name: einst (Quellcode S. 81 f.)



Name: temp1 (Quellcode S. 86 f.)



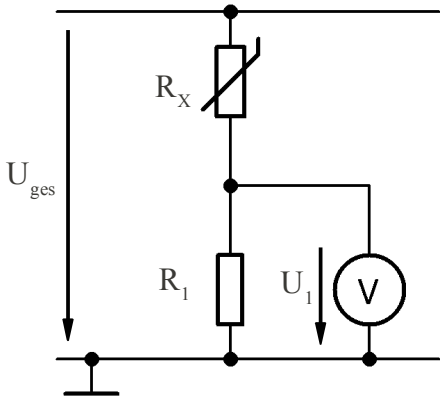
XII. 5 Protokolle

28.6.2009, 14:00	Protokoll: Widerstandsermittlung KTY I (PR01)	Johannes Höntsch
Blatt 1/1		zu Hause

Ziel: Ermittlung der Temperatur mittels KTY-10 in Vorbereitung auf die Messung via ATmega über einen Spannungsteiler.

Vorüberlegung:
- KTY-10 ist ein temperaturabhängiger Widerstand: steigt T, so steigt R_T
- Spannungsteilerregel: $\frac{U_1}{U_{ges}} = \frac{R_1}{R_1 + R_X} \rightarrow R_X = \frac{R_1 \cdot U_{ges}}{U_1} - R_1$
- Formeln für Berechnung der Temperatur im Datenblatt des KTY-10

Durchführung: Nahe des KTY-10 wird ein Temperatursensor gehängt, um die tatsächliche Umgebungstemperatur zu erhalten.
Ermittlung der elektrischen Spannung über den Sensor:



Die Spannung und Temperatur werden über 10 Minuten aller Minuten abgelesen.
Im Anschluss nach den Formeln des Datenblattes Temperatur berechnen.

$R_1 = 2250\ \Omega$
$U_{ges} = 3{,}3\ V$

Beobachtung:

t in min	0	1	2	3	4	5	6	7	8	9
U_1 in mV	1760	1759	1759	1760	1760	1759	1760	1760	1759	1760
ϑ in °C	24,0	24,0	24,0	24,1	24,1	24,1	24,1	24,1	24,1	24,1

Auswertung: Spannungsdurchschnitt: 1759,6 mV
Temperaturdurchschnitt: 24,07 °C

Somit ergeben sich nach den Formeln:
$R_X = R_T = 1970\ \Omega$
$k_T = 0{,}985$
$T = 23{,}09$ °C

Die Abweichung beträgt 0,98 °C. Sie ist gering und kann mit Messfehlern aufgrund der Kabel erklärt werden, die im mV-Bereich durchaus Auswirkungen durch Eigenwiderstand haben. Ein weiterer Grund ist der Eigenwiderstand des Sensors, der bei 2000 Ω ± 20 Ω liegt. Schon bei 1984 Ω ist das errechnete gleich dem angezeigtem Ergebnis.

Ergebnis: Der Versuch wurde erfolgreich durchgeführt.

22.7.2009, 9:45	Protokoll II a) : ADC und KTY (PR02 a))	Johannes Höntsch
Blatt 1/1		zu Hause

Ziel: Herleitung einer sensorspezifischen Temperaturfunktion in Abhängigkeit des ADC-Wertes.

Vorüberlegung:
- Messung mittels Spannungsteiler erreicht eine annähernd ideale Linearität
- allgemeine Formel: $y(x) = m \cdot x + n$
- bei Eiswasser KTY isolieren, damit er nicht überbrückt wird.

Durchführung:
- ADCW bei Zimmertemperatur ermitteln, Zimmertemperatur messen
- ADCW in Eiswasser ablesen, Eiswassertemperatur bestimmen
- Formel herleiten

Beobachtung:

ADCW	541	593
ϑ in °C	24,8	2,5

Auswertung: ϑ (ADCW) = m · ADCW + n

es entsteht ein LGS: (I) 24,8 = m · 541 + n
(II) 2,5 = m · 593 + n

$$\vartheta(ADCW) = -\frac{223}{520} \cdot ADCW + \frac{133539}{520}$$

Aus dem linearen Koeffizienten von -0,429 lässt sich ableiten, dass die Auflösung gut 0,4 °C pro Quantisierungsstufe beträgt.

Ergebnis: Es wurde erfolgreich eine Formel zur Temperaturberechnung in Abhängigkeit des ADC-Registers hergeleitet. Ihre Gültigkeit im Vergleich zur Berechnung wird in einem weiteren Protokoll geprüft. Die Abstufung ist jedoch zu groß, daher muss durch Signalverstärkung ein feinerer Messbereich erzeugt werden. Danach ist der Versuch zu wiederholen und wie geplant fortzufahren.

9.10.2009, 18:30	Protokoll II b) : ADC und KTY mit OPV (PR02 b))	Johannes Höntsch
Blatt 1/1		zu Hause

Ziel: Wiederholung des Protokolls II a) aus der Ergebnisbetrachtung: Herleitung einer schaltungsspezifischen Temperaturfunktion in Abhängigkeit des ADC-Wertes. In die Schaltung ist ein Operationsverstärker zu integrieren.

Vorüberlegung:
- Messung mittels Spannungsteiler erreicht eine annähernd ideale Linearität
- allgemeine Formel: $y(x) = m \cdot x + n$
- KTY isolieren, damit er nicht in Wasser überbrückt wird.

Durchführung:
- Operationsverstärker in Schaltung integrieren (2-fache Verstärkung)
- in heißem Wasser Temperatur und ADC-Werte ablesen
- Formel herleiten

Beobachtung:

ADCW	433	291
ϑ in °C	60	19

Auswertung: ϑ (ADCW) = m · ADCW + n

es entsteht ein LGS: (I) 60 = m · 433 + n
(II) 19 = m · 291 + n

$$\vartheta(ADCW) = 0{,}3 \cdot ADCW - 65$$

Aus dem linearen Koeffizienten von 0,3 ist ablesbar, dass die Auflösung 0,3 °C pro Quantisierungsstufe beträgt. Im Vergleich zu dem ersten Protokoll ist dies ein qualitativer Gewinn.

Ergebnis: Es wurde erfolgreich eine Formel für die Temperaturberechnung mittels ATmega hergeleitet. Die Genauigkeit wurde erhöht, ist allerdings schaltungsabhängig.

10.10.2009, 16:30	Protokoll III: ADC und KTY mit OPV (Polynom) (PR03)	Johannes Höntsch
Blatt 1/1		zu Hause

Ziel: Für die Auswertung der Temperaturmessung und die Wahl der geeigneten Berechnung soll eine Temperaturkurve in Form eines Polynoms fünften Grades in Abhängigkeit des ADCW-Wertes aufgestellt werden.

Vorüberlegung:
- es sind sechs Gleichungen notwendig
- allgemeine Form: $T(x) = ax^5+bx^4+cx^3+dx^2+ex+f$

Durchführung:
- in heißem Wasser Temperatur und ADC-Wert kontinuierlich ablesen
- Formel herleiten

Beobachtung:

ADCW	522	463	433	400	368	334	291
ϑ in °C	83	70	60	50	40	30	18,6

Auswertung: folgende Gleichungen entstehen:

$$\begin{aligned}
70 &= a\cdot 463^5+b\cdot 463^4+c\cdot 463^3+d\cdot 463^2+463\cdot e+f\\
60 &= a\cdot 433^5+b\cdot 433^4+c\cdot 433^3+d\cdot 433^2+433\cdot e+f\\
50 &= a\cdot 400^5+b\cdot 400^4+c\cdot 400^3+d\cdot 400^2+400\cdot e+f\\
40 &= a\cdot 368^5+b\cdot 368^4+c\cdot 368^3+d\cdot 368^2+368\cdot e+f\\
30 &= a\cdot 334^5+b\cdot 334^4+c\cdot 334^3+d\cdot 334^2+334\cdot e+f\\
18{,}6 &= a\cdot 291^5+b\cdot 291^4+c\cdot 291^3+d\cdot 291^2+291\cdot e+f
\end{aligned}$$

$$\vartheta(x) = 6{,}29\cdot 10^{-10}\cdot x^5-1{,}17\cdot 10^{-6}\cdot x^4+8{,}67\cdot 10^{-4}\cdot x^3-0{,}32\cdot x^2+58{,}09\cdot x-4227{,}44$$

Als Auflösung kann hier kein linearer Koeffizient angegeben werden, da keiner bei einem Polynom existiert. Wichtig ist jedoch, dass bei Temperaturen über 60 °C das Polynom nicht mehr verwendbar ist, wie Diagramm 4 zeigt.

Ergebnis: Es wurde erfolgreich eine Formel für die Temperaturberechnung mittels ATmega hergeleitet. Die Genauigkeit wurde erhöht, ist allerdings schaltungsabhängig.

Nachtrag: Die Werte Für die Herleitung wurden mit einer inzwischen angepassten Schaltung ermittelt. Daher ist das Ergebnis ein anderes Polynom, als auf Seite 41/42 angegeben.

Wegen der Herleitung ist das Polynom nur für $18{,}6 \leqslant x \leqslant 70$ zulässig und enthält ausschließlich hierfür richtige Werte.

11.10.2009, 11:00	Protokoll IV: KTY Vergleich (Polynom) (PR04)	Johannes Höntsch
Blatt 1/1		zu Hause

Ziel: Es sollen für verschiedene KTY-Sensoren das Widerstandsverhältnis KT ermittelt werden, um zu entscheiden, nach welcher Formel berechnet werden soll.

Vorüberlegung:
- jeder Sensor hat herstellungsbedingt einen anderen Widerstand
- da immer die gleiche Formel angewandt werden muss, muss auch der Divisor für k_T gleich sein
  → es reicht aus, nur R_{KTY} zu betrachten, da k_T nur das Verhältnis zu 1 kΩ bildet (verwendeter Typ besitzt anderen Grundwiderstand vgl. S. 42, 97)

Durchführung:
- es werden fünf KTY (81 120) bei gleicher Temperatur nach ihrem Widerstand ausgemessen
- Ergebnisse vergleichen

Beobachtung:

KTY	1	2	3	4	5
R_{KTY} in Ω	960	968	975	962	967

Auswertung:

$$\bar{x} = \mu = \frac{960\,\Omega + 968\,\Omega + 975\,\Omega + 962\,\Omega + 967\,\Omega}{5} = 966{,}4\,\Omega$$

$$\sigma = \sqrt{\frac{(960\,\Omega - \mu)^2 + (968\,\Omega - \mu)^2 + (975\,\Omega - \mu)^2 + (962\,\Omega - \mu)^2 + (967\,\Omega - \mu)^2}{5}} = 5{,}238\,\Omega$$

Das arithmetische Mittel liegt bei $\bar{x} = \mu = 966{,}4\,\Omega$. Daraus ergibt sich eine Standardabweichung von $\sigma = 5{,}238\,\Omega$. Also liegt der Widerstand der Messreihe bei $R_{ges} = 966{,}4\,\Omega \pm 5{,}2\,\Omega = 966{,}4\,\Omega \pm 0{,}54\,\%$.

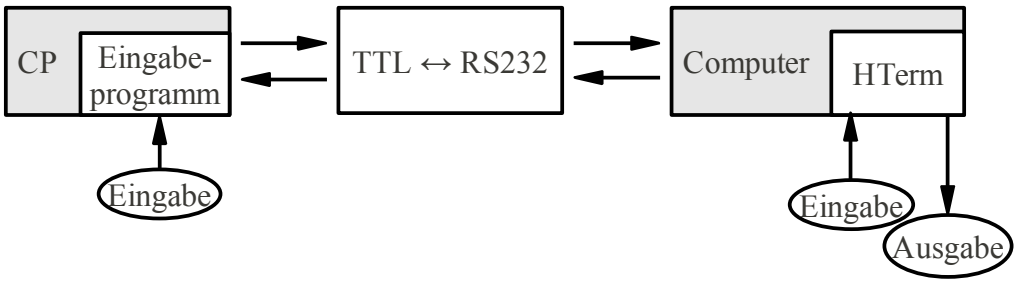
Ergebnis: Die Werte differieren um gut 0,5 %. Das ist eine geringe, aber vorhandene Abweichung. Es ist zudem zu beachten, dass diese statistische Erhebung auf einem Umfang von nur fünf Proben beruht.

12.10.2009, 16:30	Protokoll V: Datenverkehr bei der Kommunikation von ClassPads (PR05)	Johannes Höntsch
Blatt 1/3		zu Hause

Ziel: Um die Kommunikation von µC und ClassPad zu ermöglichen, muss das Kommunikationsprotokoll in den µC implementiert werden. Es ist herauszufinden, wie das Protokoll aufgebaut ist, die Zeichen gesendet werden und welche Steuercodes welche Bedeutung haben.

Vorüberlegung: Die Kommunikation ist seriell, d.h. es gibt eine Datensende- und eine Datenempfangsleitung, die zwischen Datenquelle und -senke gekreuzt wird. Um den Pegel zu ermitteln, verfügen beide über eine gemeinsame Masseleitung.
Es kann immer nur eine Datenrichtung abgehört werden.
Die Kommunikationsparameter (Baudrate, Stopbits...) müssen bei Sender und Empfänger gleich sein.

Durchführung:



- Verkabelung durchführen
- Programme starten
- vom CP ein Zeichen senden, antworten und Datenverkehr notieren

Beobachtung:

die Zahl „856“ wird gesendet:

Bezeichnung	**Richtung (CP | PC)**	**Zeichen (hex)**
Handshakeanfrage	→	15
Handshakeantwort	←	13
NDd-Header	→	3A | 4E | 44 | 64 | 00 | 01 | 00 | 01 | 00
NDd-Bestätigung	←	06
Zahl senden	→	3A | 00 | 01 | **38** | **35** | **36** | 00 | *5C*
Zahl bestätigen	←	06

Weitere Sendevorgänge zur Analyse (es wird jeweils nur der Datenstrom des „Zahl senden“ abgebildet, sofern NDd-Header gleich sind):

Zahl	**gesendete Zeichen**
857	3A | 00 | 01 | **38** | **35** | **37** | 00 | *5B*
858	3A | 00 | 01 | **38** | **35** | **38** | 00 | *5A*
1	3A | 00 | 01 | **31** | 00 | 00 | 00 | *CE*
0	3A | 00 | 01 | **30** | 00 | 00 | 00 | *CF*
100	3A | 00 | 01 | **31** | **30** | **30** | 00 | *6E*
999	3A | 00 | 01 | **39** | **39** | **39** | 00 | *54*

Bei folgenden Beispielen änderte sich der NDd-Header:

neuer NDd-Header: 3A | 4E | 44 | 64 | 00 | 01 | 00 | 01 | 00 | 0A | 00 | 0A | 05 | FF | F0

Zahl	**gesendete Zeichen**
1023	3A | 00 | 01 | **31** | **30** | **32** | **33** | 00 | 00 | 00 | 00 | *39*
1000	3A | 00 | 01 | **31** | **30** | **30** | **30** | 00 | 00 | 00 | 00 | *3E*
1002	3A | 00 | 01 | **31** | **30** | **30** | **32** | 00 | 00 | 00 | 00 | *3C*

Auswertung:

Das Protokoll ist folgendermaßen aufgebaut:

Handshake	NDD-Header	Datenstrom
Anfrage - Bestätigung 0x15 - 0x13	NDD - Bestätigung 0x3A... - 0x06	Datenkopf, Daten, 0 Prüfsumme - Bestätigung 0x3A..., 0x30..., 0x00 0xyy - 0x06

Die Bestätigung des Handshakes erfolgt mit *0x13*, die aller anderen Daten mit *0x06*. NDd-Header und Datenkopf haben das gleiche Anfangszeichen *0x3A*.

Zwischen dem letzten Zeichen des Datenstromes und der gesendeten Zahl ist mindestens eine Leerstelle (*0x00*). Es muss also zwischen der letzten Zahl und dem letzten Zeichen des Datenstromes ein Null-Zeichen eingefügt werden.

Die Zahl wird in ihre einzelnen Ziffern zerlegt, welche dann im ASCII-Format gesendet werden (*358* → 0x33 | 0x35 | 0x38).

Die letzte Stelle im Datenstrom ist immer anders, wenn sich die gesendeten Zeichen ändern. Dies wird die Prüfsumme sein, welche immer neu generiert wird.

Das Schema bleibt immer gleich, lediglich der NDd-Header einer vierstelligenstelligen Zahl ist um sechs Zeichen länger als der einer dreistelligen Zahl. NDd-Header ändern sich sonst nicht.

Wird also eine Zahl mit bis zu drei Stellen gesendet, so wird der kurze NDd-Header verwendet und nach einem Nullzeichen die Prüfsumme angehängt. Bei einer vierstelligen Zahl wird der längere NDd-Header verwendet, da offenbar nur vier Zeichen (mit Nullzeichen) zwischen Datenkopf und Prüfsumme stehen können. Bei dem längeren NDd-Header sind es mit Nullzeichen sechs Ziffern, die gesendet werden können.

Da die Umrechnung in eine Temperatur im ClassPad stattfindet, müssen vom µC mittels UART nur Zahlen von 0 bis 1023 übertragen werden können. Es sind also keine anderen Zahlenlängen oder gar Buchstaben für die Übertragung nötig und deren Übermittlung muss nicht analysiert werden.

Ergebnis: Der Versuch wurde erfolgreich durchgeführt, das Kommunikationsprotokoll wurde analysiert und ausgewertet.

Es ergeben sich folgende weitere Schritte für die Emulation mittels µC:

- Handshake integrieren
- NDd-Header je nach Länge der Zahl (3 oder 4 Stellen) ausgeben
- Dateikopf versenden
- Zahl in Ziffern zerlegen und diese im ASCII-Format senden
- Nullzeichen einfügen (ggf. mehrere)
- Prüfsumme berechnen und anhängen

Die ersten Schritte stellen programmiertechnisch kein Problem dar, da die Ausgabe von Zeichen mit UART bereits integriert ist. Ebenso kann einfach festgestellt werden, wie lang die Zahl ist. Daraus abgeleitet kann sie in ihre Ziffern zerlegt werden.
Nun ist zu recherchieren, wie sich die Prüfsumme nachbilden lässt.

19.02.2010, 15:15	Protokoll VI a): Vergleich zwischen EA-200 und CPAD (PR06 a))	Johannes Höntsch
Blatt 1/2		zu Hause

Ziel: Es ist anhand von aufgestellten Kriterien ein Vergleich zwischen dem von CASIO hergestellten Messerfassungssystem und dem in der BELL entwickelten Produkt durchzuführen und auszuwerten.

Vorüberlegungen: Von CASIO gibt es das ClassPad Add-In *ECON*. Im Handbuch des EA-200 gibt es aber zudem Programmbeispiele u.a. für den ClassPad. Diese zwei Erfassungsmöglichkeiten sind – falls Unterschiede bestehen – gesondert zu betrachten (CASIO: Programmbeispiele, ECON: Add-In von CASIO).

Durchführung:

- Hardware parallel testen
- spezielle Eigenschaften einzeln erfassen
- Ergebnis auswerten

Beobachtung:

Kriterium	**EA-200** [1]	**CPAD**
Messschnelligkeit (bezogen auf Echtzeit)	CASIO: k.A. ECON [2]: 0,3 … 299 s	2 … 999 s
Handbuch	Ja; bebildert und Beispiele, Befehlsreferenz → selbst Programm schreiben möglich	Ja; bebildert, allgemeiner Aufbau für Messung; einzelne Programmpunkte ausführlich erklärt
Statusmeldungen an Hardware (LEDs)	I/O, Bereit, Messaktivität, Fehler, Batterie	I/O, Messtyp/-aktivität, Funkstatus (Senden, Empfangen, Bereitschaft)
Anwendungscharakter	Schüler/Lehrer in Kombination	Schüler/Lehrer einzeln
Energieversorgung	4 AA-Batterien; Netzteil	3 AAA-Batterien
Sensorik (Standard)	Temperatur (–20 … +130 °C), Licht, Spannung	Temperatur (–30 … +130 °C) bzw. Widerstand
Sensorik (Optional)	Beschleunigung, Druck	Strom/Spannung, Druck, Licht (einzelne Farbwerte), Beschleunigung, …
Kompatibilität	ClassPad, Algebra FX, CFX-9850G/fx-7400G	ClassPad; andere möglich → Umprogrammieren erforderlich
Programm	CASIO: Programmbeispiele im Handbuch (auch Download) ECON [2]: Add-In (nur ClassPad)	bei Temperaturmessung alles vorhanden; andere Sensoren nachprogrammierbar
Menüführung/ Nutzerfreundlichkeit	CASIO: keine ECON [2]: vorhanden/Informationen über Fehler und -behebungsansätze	Ja, kleine Hilfeanzeigen

19.02.2010, 15:15	Protokoll VI a): Vergleich zwischen EA-200 und CPAD (PR06 a))	Johannes Höntsch
Blatt 2/2		zu Hause

Kriterium		**EA-200**	**CPAD**
Auswerteoptionen		Regressionen, graphischer Verlauf der Messwerte und approximierten Funktion, statistische Kalkulationen (Statistik-Menü)	
		(ECON [2]: standardmäßig keine)	
Echtzeit		wählbar	Ja
Datenübertragung		Kabel (30 cm)	Funk (bis zu 5 m bei Standardantenne)
Messdatenberechnung		fest	einstellbar (Polynom) → Datenmanipulation für Regression möglich
Preis	Hardware	150 € [1]	ca. 25 €
	Software	CASIO: 0 € ECON [2]: 0 €	0 €

[1] Informationen lt. CASIO: EA-200 Bedienungsanleitung. RJA519569-001V01
[2] ECON v1.02
[3] lt. http://www.dynatech.de/produkte/produkt.php?prod=2322; 12.2.2010, 11:00

zu Messgenauigkeit s. PR06 b)

Auswertung:

Beide Systeme haben Vor- und Nachteile. Für den schnellen Einsatz im Unterricht ist das EA-200 mit ECON sehr gut geeignet. Es ermöglicht Echtzeitmessungen in 0,3 s-Schritten. Die von CASIO angebotenen Programme sind nicht sehr benutzerfreundlich, ermöglichen aber Messintervalle von 0,00002 s. Dies ist möglich, da die Daten erfasst und erst nach der Messung gesendet werden (nicht Echtzeit). Das entwickelte Produkt positioniert sich in der Software mit einer anwenderfreundlichen Oberfläche und vielen Auswertemöglichkeiten zwischen dem von CASIO und ECON. Die Möglichkeiten der Messwerterfassung liegen zwar hinter dem EA-200, allerdings sind sie für Experimente im Bereich der Datenkommunikation und für selbstständiges Erweitern (Experimentierplatine) bestens geeignet. Zudem ist es um einiges günstiger als das EA-200.

Ergebnis:

Das entwickelte Produkt hat die anvisierten Kriterien erfüllt. Zudem geschieht die Standard-Datenübertragung per Funk, was beim EA-200 nicht der Fall ist.

19.02.2010, 11:45	Protokoll VI b): Vergleich zwischen EA-200 und CPAD (PR06 b))	Johannes Höntsch
Blatt 1/2		zu Hause

Ziel: Messgenauigkeit und -differenzen der Geräte ermitteln

Geräte/Substanzen:

- Becherglas
- CPAD, EA-200, 2 ClassPads
- Laborthermometer
- Wasser
- Schnee
- Strohhalm (als offene Pipette)

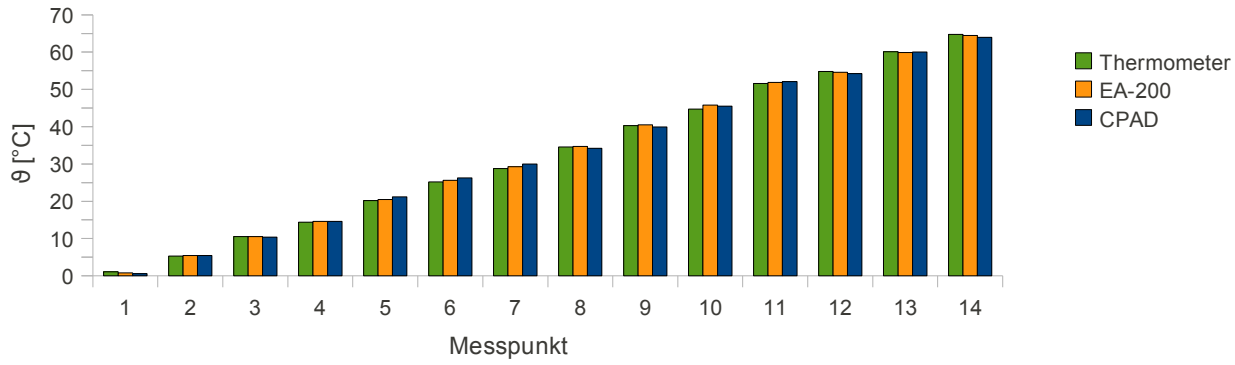
Durchführung:

- ca. aller 5 °C Werte ermitteln
- Wasser erhitzen, Wert ermitteln
- Wasser mit kälterem abkühlen, Wert ermitteln (bei Bedarf wieder warmes Wasser zugeben)
- CPAD, Laborthermometer und EA-200 gleichzeitig vergleichen
- von 0 °C bis 15 °C Schnee zur Abkühlung hinzugeben

Beobachtungen:

Messung	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Therm. [°C]	1,1	5,3	10,5	14,4	20,2	25,2	28,8	34,6	40,3	44,7	51,6	54,8	60,1	64,8
EA-200 [°C]	0,8	5,4	10,5	14,6	20,5	25,6	29,3	34,7	40,5	45,8	51,9	54,6	59,9	64,5
CPAD [°C]	0,6	5,4	10,4	14,6	21,2	26,3	30,0	34,2	39,9	45,5	52,1	54,2	60,0	64,0

Auswertung:



Wie das Diagramm zeigt, sind alle Messwerte eines Messpunktes relativ nahe beieinander.

$$\mu_{Thermometer} = 32{,}6\ °C \qquad \Delta(\mu_{ges}; \mu_{Thermometer}) = 0{,}1\ °C$$
$$\mu_{EA-200} = 32{,}76\ °C \qquad \Delta(\mu_{ges}; \mu_{EA-200}) = 0{,}06\ °C$$
$$\mu_{CPAD} = 32{,}74\ °C \qquad \Delta(\mu_{ges}; \mu_{CPAD}) = 0{,}04\ °C$$
$$\mu_{ges} = 32{,}7$$

19.02.2010, 11:45	Protokoll VI b): Vergleich zwischen EA-200 und CPAD (PR06 b))	Johannes Höntsch
Blatt 2/2		zu Hause

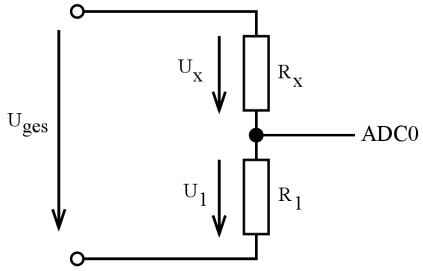
Ergebnis: Wie die Berechnung der Mittelwerte zeigt, weicht der der entwickelten Funkmesseinheit am wenigsten vom Gesamtmittelwert ab. Das ist ein mehr als erwartet gutes Ergebnis. Natürlich kann es nicht allein für die Messgenauigkeit verwendet werden. In der Gesamtheit weicht der Messwert des CPAD größer von den zwei anderen ab. Da dies in positiver und negativer Richtung geschieht, gleicht es sich aus.

Im Wesentlichen sind die Differenzen aber unter 1 °C. Zudem sollen die Experimente, bei denen das CPAD zur Anwendung kommen kann, eher Tendenzen zeigen, als hoch genaue Temperaturermittlung ermöglichen.

Der Test kann durchaus als positiv gewertet werden.

XII. 6 Schaltpläne

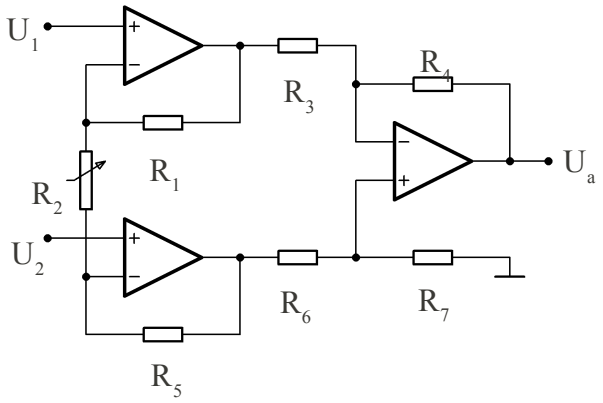
SL01 – Widerstand per AVR messen



Es gilt: $\frac{U_1}{U_{ges}} = \frac{R_1}{R_1 + R_X}$

für die Bestimmung von R_X

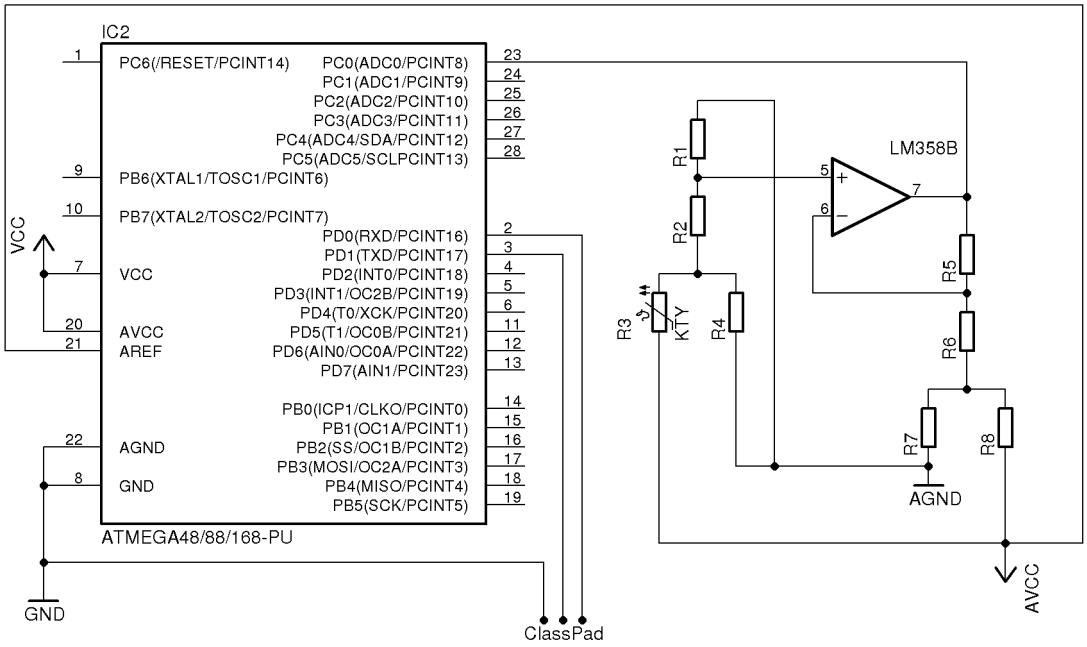
SL02 – Differenzverstärker[2]



für $R_1 = R_5$ und
$R_3 = R_4 = R_6 = R_7$ gilt:

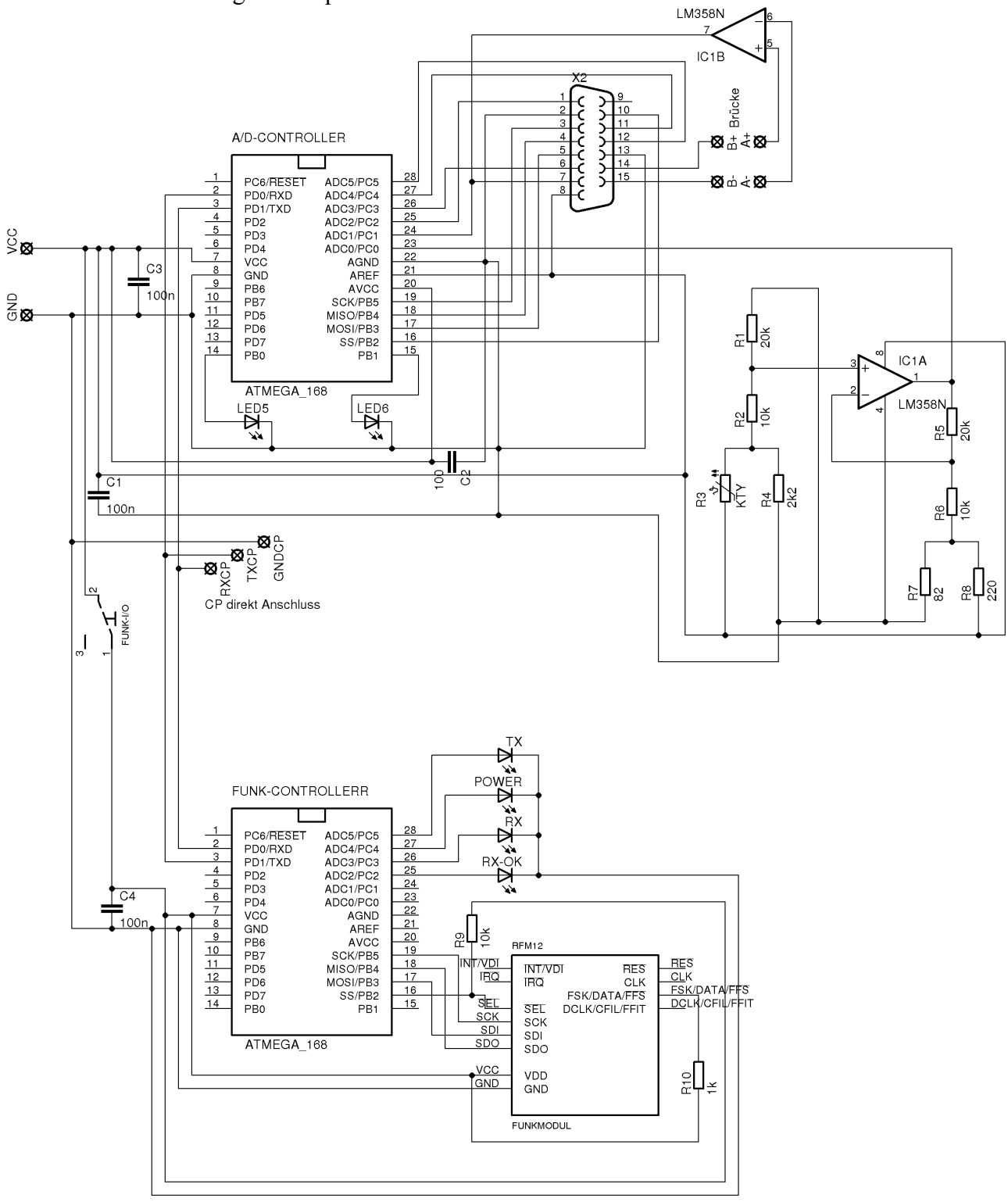
$$U_a = (1 + \frac{2 \cdot R_1}{R_2}) \cdot (U_2 - U_1)$$

SL03 – Temperaturmessung und -ausgabe per UART an einem ClassPad/Computer



2 vgl. [Q3], S. 19 - „High Input Z Adjustable-Gain DC Instrumentation Amplifier“

SL04 – Der fertige Schaltplan



XII. 7 Sonstige Anhänge

S01 – verwendete Software

Grundsystem: openSUSE 11.0 (2.6.27.25-0.1-pae); KDE 3.5.10 (release 21.12.1)

3D-Animationen:	Platine: Eagle3D 1.05 und POVRay 3.5 sonstige: Houdini Master 10.0.528 (Non-Commercial Edition)
Bildbearbeitung:	GIMP 2.6.2
Diagramme:	OpenOffice.org Calc 3.0.0 – openSUSE (9319)
Formeln:	OpenOffce.org Math 3.0.0 – openSUSE (9319)
Layout s. Schaltplan	
Mindmap:	VYM – View Your mind 1.12.2
PAP:	OpenOffice.org Draw 3.0.0 – openSUSE (9319)
Präsentation:	OpenOffice.org Impress 3.0.0. – openSUSE (9319)

Programmierung

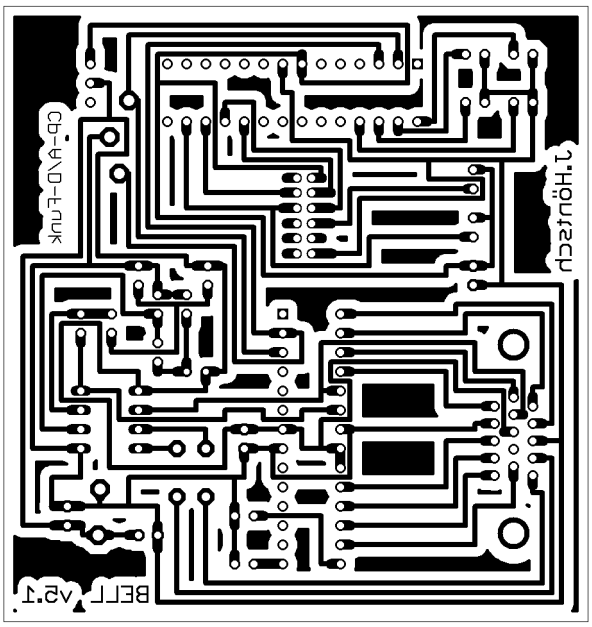
AVR-C
- Editor: KWrite 4.5.10
- Compiler: avr-gcc 4.1.3 20080612-26.5 (prerelease) (SUSE Linux)
- Flashen: PonyProg 2000 v2.07a

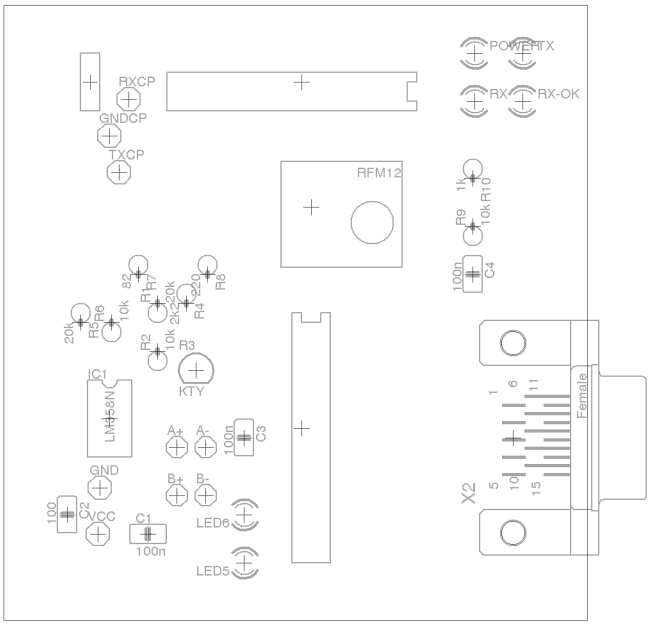
C (Vorprogrammierung der Testroutinen für ATmega, C-Experimente)
- Editor: KWrite 4.5.10
- Compiler: gcc 4.3-34.168 (UNIX)

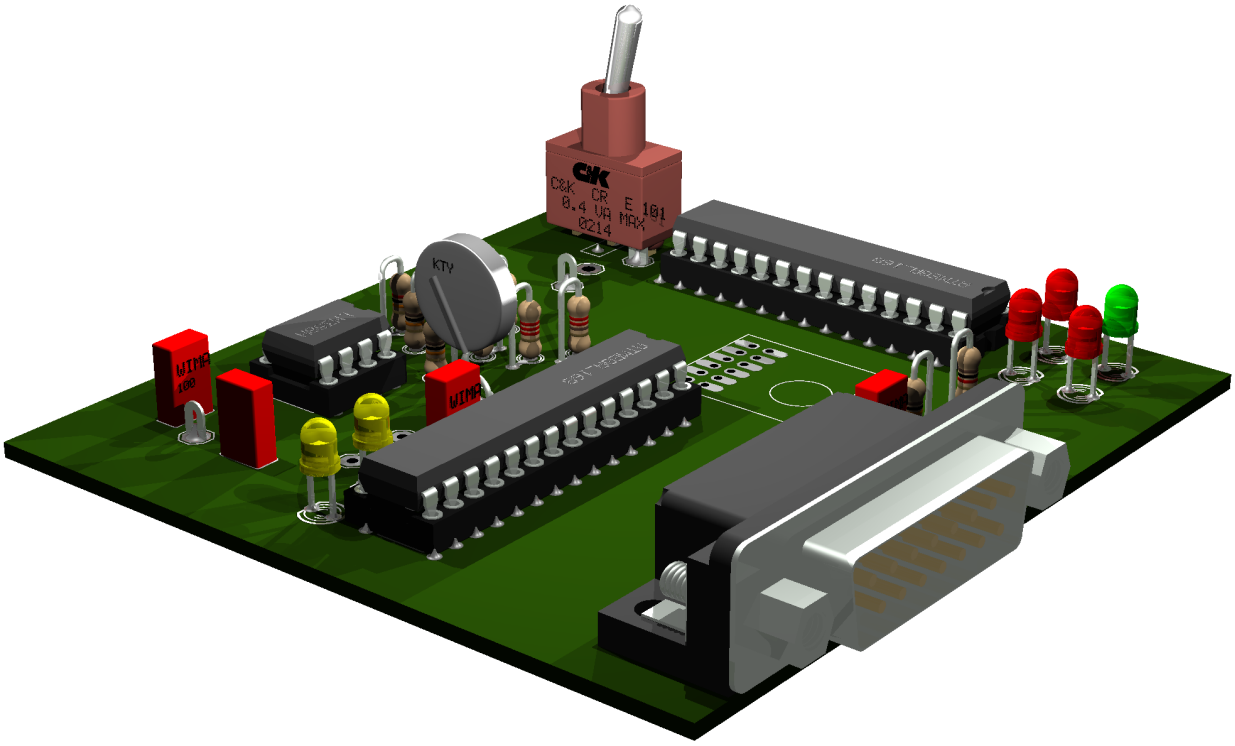
ClassPad
- ClassPad Manager 300 PLUS (Programmteiletests)
- ClassPad 300 PLUS, OS 3.04.4000

Schaltplan:	EAGLE Light, Version 6.5.0 (Linux)
seriellen Port auslesen:	HTerm 0.7.1 beta
Technische Zeichnungen:	QCad v2.0.5.0; 23.5.2009
Textproduktion:	OpenOffice.org Writer 3.0.0 – openSUSE (9319)

S02 – Layout, Bestückungsplan und 3D-Platine



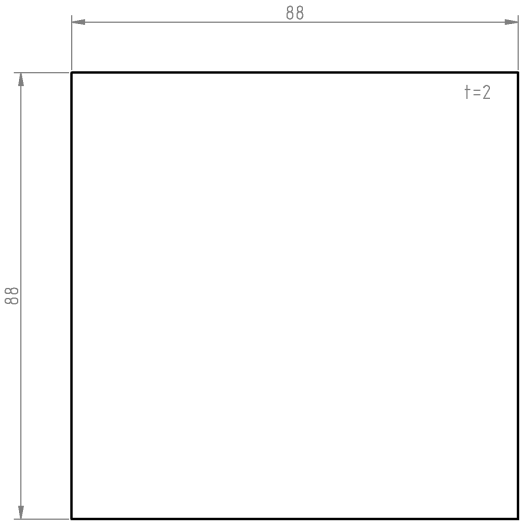




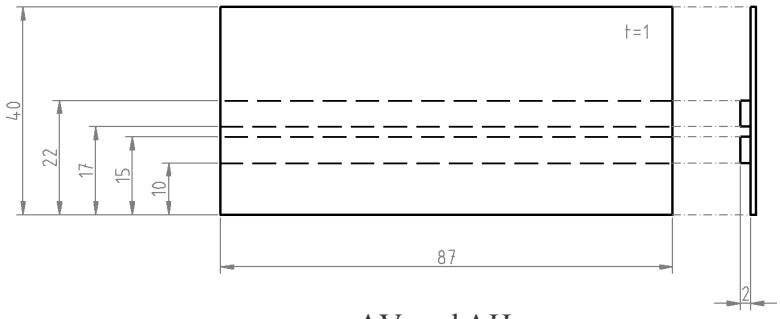
Befehl zum Rendern des Bildes: `povray schalt_5_v2.pov +w2048 +h1536`

Kameraachse: $\vec{x} = \begin{pmatrix} 150 \\ 109 \\ -186 \end{pmatrix} + r \begin{pmatrix} -150 \\ -127 \\ 186 \end{pmatrix}$, $\alpha = 25°$

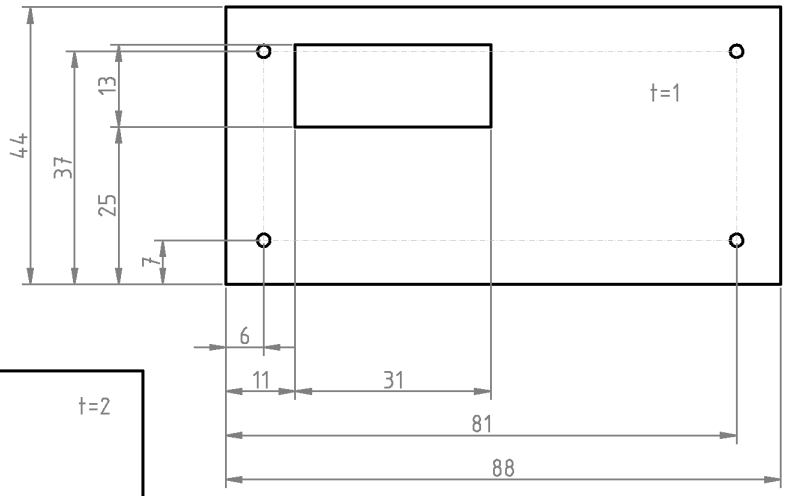
S03 – Gehäuse: CAD-Zeichnungen und 3D-Bild



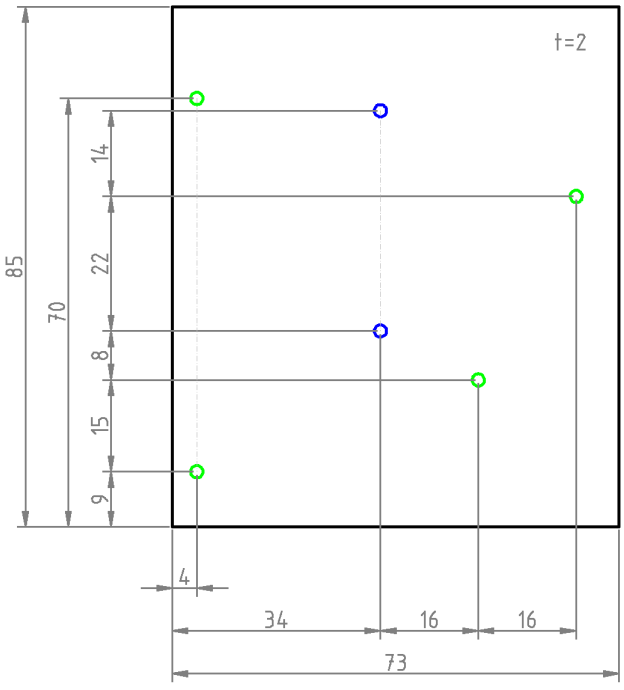
AU



AV und AH



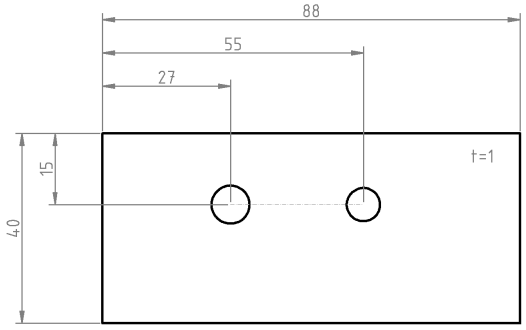
AR



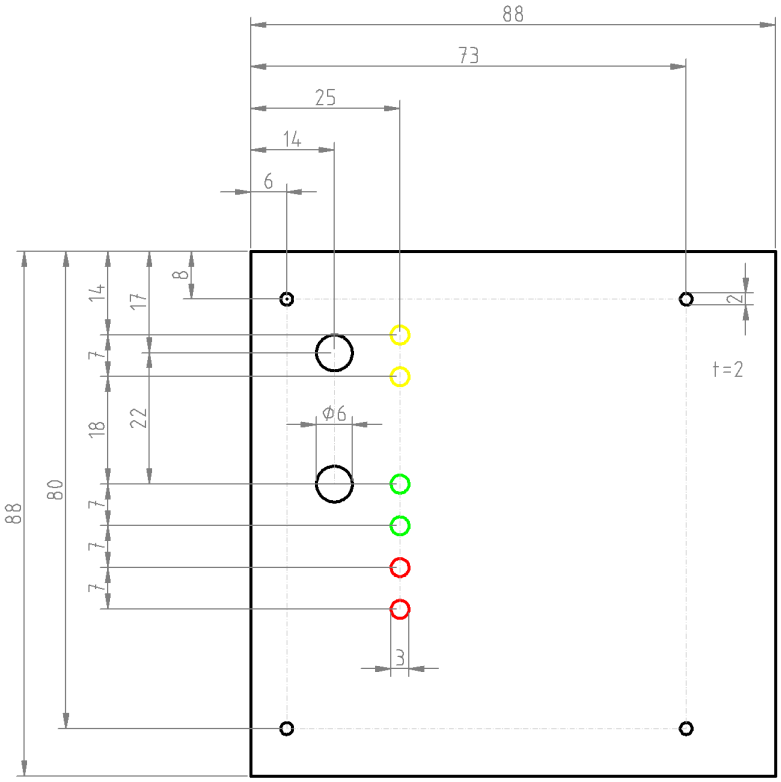
Mittelplatte

Platienenborung

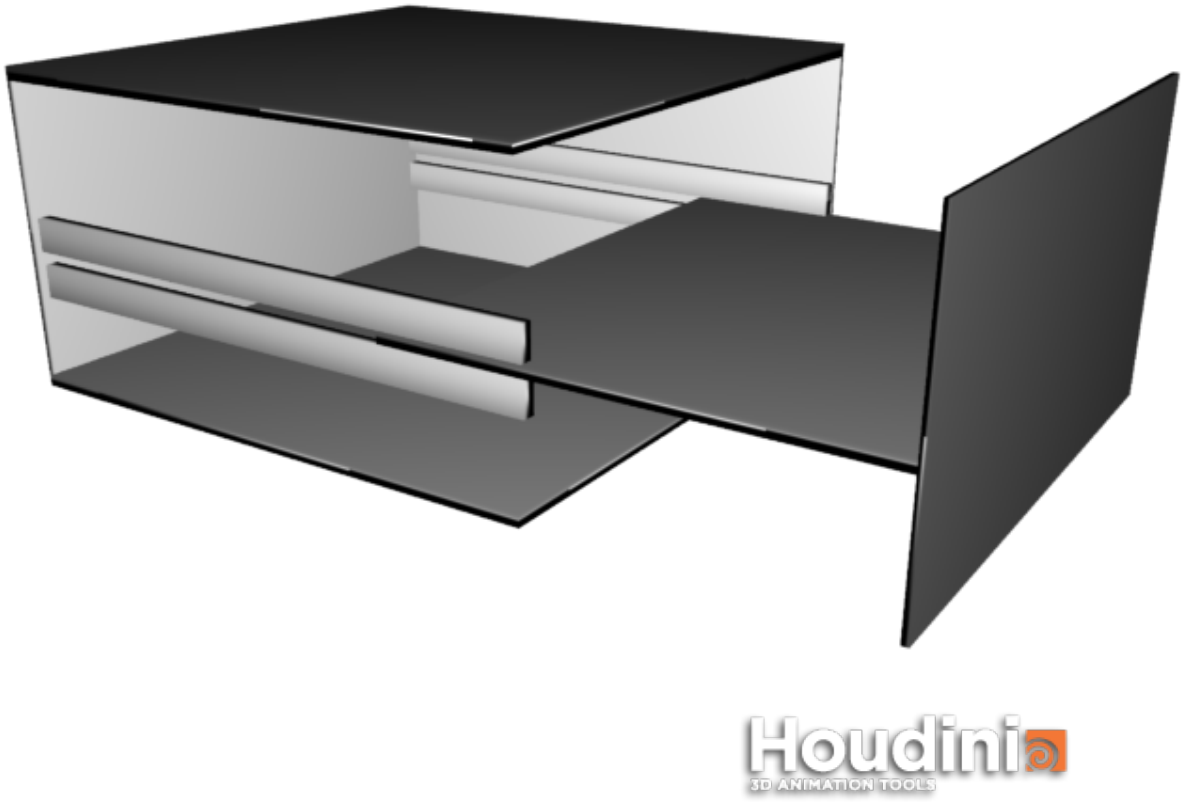
Batteriefachbohrung



AL


88
73
25
14
6
8
14
17
7
7
18
22
88
80
7
7
7
ϕ6
3
2
t=2

AO


Houdini
3D ANIMATION TOOLS

S04 – Kostenaufstellung für CPAD

Kategorie	Bezeichnung	Stk.	Bezug	Stk. €	Preis
Gehäuse	Hartpapiermontageplatte 100x200x1	1	Conrad	1,44 €	1,44 €
	Hartpapiermontageplatte 200x250x2	1	Conrad	2,78 €	2,78 €
	Schrauben M2, 100mm	8	-	0,00 €	0,00 €
	Muttern M2	8	-	0,00 €	0,00 €
Elektronik	Schalter	2	(CSD Electronics)	0,35 €	0,75 €
	LED grün 3mm	2	(Conrad)	0,08 €	0,16 €
	LED rot 3mm	2	(Conrad)	0,08 €	0,16 €
	LED gelb 3mm	2	(Conrad)	0,08 €	0,16 €
Platine	fotopositive Epoxyd-Platine 160x100	0,5	Conrad	2,93 €	1,47 €
Bauteil	Widerstand Metallschicht 2,2 kΩ	1	(Conrad)	0,09 €	0,09 €
	Widerstand Metallschicht 1 kΩ	1	(Conrad)	0,09 €	0,09 €
	Widerstand Metallschicht 10 kΩ	3	(Conrad)	0,09 €	0,09 €
	Widerstand Metallschicht 20 kΩ	2	(Reichelt)	0,08 €	0,08 €
	Widerstand Kohleschicht 220 Ω	1	(Conrad)	0,10 €	0,10 €
	Widerstand Kohleschicht 82 Ω	1	(Conrad)	0,10 €	0,10 €
	KTY-81	1	Reichelt Elektronik	0,55 €	0,55 €
	Keramikkondensator 100nF	4	(Conrad)	0,22 €	0,88 €
	Sockel 28-polig	2	(Conrad)	0,35 €	0,70 €
	Sockel 8-polig	1	(Conrad)	0,13 €	0,13 €
	Buchsenleiste 2x8-polig gerade, 2mm	1	(IT-WNS)	0,69 €	0,69 €
IC	AVR ATmega168 PDIP	2	(CSD Electronics)	2,95 €	5,90 €
	LM 358	1	IT-WNS	0,12 €	0,12 €
	RFM-12-B	1	(IT-WNS)	4,29 €	4,29 €
Anschluss	D-Sub 15polig, gewinkelt	1	Conrad	2,23 €	2,23 €
	Stereo Klinkenbuchse 2,5mm	1	Conrad	0,55 €	0,55 €

Summe: **23,46 €**

Bei Bezugsquellen in Klammern war das Produkt bereits vorhanden, der gelistete Preis bezieht sich auf die angegebene Bezugsquelle.

S05 – Projektantrag

BSZ Bau und Technik Dresden

Berufliches Gymnasium Fachrichtung Technik

Antrag für Belegarbeit / Facharbeit / BELL[1]

Name, Vorname	**Jahrgang** Bgy07
Höntsch, Johannes	**Fach / Kurs** **Datenverarbeitungstechnik** Leistungskurs

Thema der Arbeit

ClassPad 300 PLUS-Anwendungen im Chemieunterricht

Kurze Themenbeschreibung

Es werden geeignete Unterrichtsszenarien für den Chemieunterricht unter Einbeziehung des ClassPad-Systems hergeleitet.

Für diese Unterrichtsszenarien werden Hard- und Software entwickelt und mit dem ClassPad-System verbunden.

Die Gebrauchstauglichkeit für den Chemieunterricht wird im Rahmen einer Nutzerakzeptanzanalyse nachgewiesen.

Termine	
Beginn der Arbeit:	24. August 2009
1. Konsultation:	12. Oktober 2009
2. Konsultation:	23. Oktober 2009
3. Konsultation:	9. Februar 2010
Abgabe der Arbeit:	26. Februar 2010
Verteidigung:	3. Juni 2010

Betreuender Lehrer

Name:	Herr Große
Büro:	E17
Postanschrift:	BSZ Bau und Technik Dresden Güntzstraße 3-5 01069 Dresden

Arbeitsumfeld

BSZ Bau und Technik Fachbereich BGy – Lernbereich Schule

Arbeitsphasen (einschließlich Zeitplanung)

1 Zutreffendes unterstreichen

Herleitung der Unterrichtszenarien Finden von Messanwendungen und Einordnung in den Lehrplan Beschreibung und Anwendung des Unterrichtsmittels ClassPad300+ IST-Analyse zu Funkmodul für ClassPad300+ SOLL-Analyse für Messeinrichtung und Auswertung mittels ClassPad300+	**8 Std.**
Dokumentation Ziel und Zweck der Arbeit, Vorgehensweise IST-Analyse SOLL-Analyse	**18 Std.**
Entwurf Messsystem und Anwendungsszenarien Entwurf Hardware und Schnittstellen Entwurf Software mit Daten, Oberfläche, Funktionen für PC und ClassPad300+ Entwurf Anleitung für den Unterricht, Parameter zur Einsatzbegrenzung	**80 Std.**
Implementierung und Integration Umsetzung der Entwürfe und Komponententests Messsystemtests vornehmen Anleitungen für den Unterricht anpassen Nutzerakzeptanzanalyse durchführen, auswerten und Ergebnisse einarbeiten	**120 Std.**
Dokumentation Zusammenfassung Implementierung und Integration Ergebnisse des Messsystemtests und der Akzeptanzanalyse Fazit und Ausblick zur Arbeit	**60 Std.**

Geforderte Dokumentation(en) zur Arbeit

BELL (wissenschaftliche Arbeit)
Anleitungen für den Unterricht (Anlage zur BELL)
Konstruktions- und Schaltungsunterlagen zur Hardware (Anlage zur BELL)
Software (CD-Version als Anlage zur BELL)

BELL ist als **dreifache** gebundene Papierversion abzugeben,
Tools und Anleitungen entsprechend in Papier- bzw. elektronischer Version.

Benötigte Präsentationsmittel[2]

☐ Flipchartchart	☒ Overhadprojektor	☐ Pinwand
☒ PC, Beamer	☒ Sonstiges: OH-ClassPad Set, Tafel bzw. Whiteboard	

Antragsteller	**Bestätigung des betreuenden Lehrers**
Dresden, 24. August 2009	Dresden, 24. August 2009
…… Unterschrift	…… Unterschrift

2 Zutreffendes ankreuzen

XIII. Selbstständigkeitserklärung

Dresden, den 22.2.2010

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Johannes Höntsch