F15 - Anleitung: Messwerterfassung und Prozesssteuerung

Sven Brieden 22.10.2018

Vorwort zum Versuch

Im Verlauf des Versuchs werden Sie einige Prozesse zur Aufnahme von Messgrößen erleben sowie Strategien kennenlernen und selbst entwickeln, um externe Parameter zu kontrollieren.

Solche Prozesse sind oft mit vielen einzelnen Schritten verbunden (Kalibrierungen, Referenzierungen,etc.) die erst sauber durchgeführt werden müssen, bevor zuverlässige Ergebnisse generiert werden können.

Ziel des heutigen Versuchs wird es sein, Ihnen solche Prozesse an diversen Beispielen aufzuzeigen und Schritt für Schritt aus unbekannten Signalen belastbare Daten zu generieren. Der Ablauf und die Sequenz der einzelnen Versuchsteile orientieren sich an dem, was bei der Vorbereitung eines komplexen Forschungsexperiments real auftritt.

Inhaltsverzeichnis

Einordnung des Versuchs

Zum Kennenlernen der Arbeit im Labor wurden im Grundpraktikum generell alle Messgeräte manuelle angesteuert und alle Messergebnisse manuelle aufgenommen. Dadurch diese stark verringerte Komplexität der Versuche kann der Fokus

auf Grundlagen gelenkt: Auswertung der Messdaten, sowie deren kritische Einordnung und die Fehleranalyse. Die Praxis im Fortgeschrittenenpraktikum besteht aus Teilautomatisierung und detaillierten Parameter-Scans. Dabei wird meistens vorrausgesetzt, dass die Messwerterfassung und Prozesssteuerung ohne Probleme funktioniert. Diese „Black Box“ versuchen wir in diesem Versuch zu öffnen.

Messwerterfassung

Die einzelnen Verarbeitungsschritte der Messwerterfassung sind: 1. physikalischer Prozess, der analysiert werden soll 2. Sensor registriert eine physikalischen Größe und liefert häufig ein Spannungssignal. Dieses Signal ist im einfachsten Fall analog(an/aus) beispielsweise von einer Lichtschranke. Viele Temperatursensoren und Drucksensoren haben ein linearen Zusammenhang zwischen der Messgröße und dem Ausgangssignal. Sowohl die Art des Ausgangssignals als auch die mit der Messgröße verbundene Zusammenhang wird varriabel auf das aktuelle Probelm angepasst. 3. Signalaufbereitung, verbessert das Signal durch bespielsweise Rauschentfernung. Es wird für den folgenden Schritt der Analog-Digital Wandlung optimiert. 4. Analog-Digital Wandler macht aus dem aufbereiteten Ausgangssignal des Sensors ein für den PC und die standard Verarbeitungswege verständliches Signal. 5. Digitale Verarbeitung ist meistens der letzte Schritt, bei dem das Signal gesammelt, dargestellt und ausgewertet wird. Dabei können die Signale auf die Messgrößen zurückgeführt werden und mittels Messwertauswertung verglichen werden.

Programmierung

Datentypen

Im Folgenden werden die elementaren Datentypen vorgestellt, welche wir in der Programmierung, im Besonderen bei der Arbeit mit Variablen brauchen. Datentypen dienen der besseren Ordnung in einem Programm und ermöglichen eine effektive Nutzung des Speicherplatzes.

Bei der Wahl des richtigen Datentyps spielen folgende Überlegungen eine wichtige Rolle: - Was möchte ich speichern oder verarbeiten? - Zeichen - Text - Zahl - Kommazahl

Datentypen haben unterschiedliche Verwendungsformen. So kann der Datentyp ganze Zahl z.B. keine Buchstaben und keine Kommazahlen speichern, sondern nur ganze Zahlen. Datentypen haben unterschiedliche Wertebereiche. Der Wertebereich sagt aus, welche Zahlen dieser Datentyp speichern kann. Je nach dem, ob man eher kleine Zahlen bis 65535, etwas größere bis 4.294.967.295 oder ganz große Zahlen mit 20 Stellen speichern muss, wird der Datentyp entsprechend ausgesucht.

Ganze Zahlen: int

Mit dem Schlüsselwort int erstellen wir Variablen gewöhnlicher Größe. Diese Variante wird auch am Meisten verwendet. Laut Standard hat dieser Datentyp mindestens 16 Bit, bei einem 32-Bit Prozessor jedoch 32 Bit. Daraus ergibt sich ein Wertebereich von -2.147.483.647 bis +2.147.483.647, bei fehlendem Vorzeichen von 0 bis 4.294.967.295.

Kommazahlen

Wie bei den ganzen Zahlen gibt es bei den Kommazahlen, auch Fließkommazahlen genannt, unterschiedliche Varianten – je nach benötigter Größe: - float mit 32 Bit - double mit 64 Bit - long double mit 80 Bit

Die genauen Grenzen der Wertebereiche sind von System zu System unterschiedlich. Für diesen Versuch ist die Genauigkeit und der Wertebereich einer float als Kommazahl ausreichend. Der Wertebereich lässt sich durch die betragsmäßig größte Zahl von 3*10^38 und einer Genauigkeit bis zur 7 Dezimalstelle schreiben.

Deklaration von Variabeln

```
// Variablen deklarieren
int zahl, andere_integer;
float kleine_zahl;
...
```

Mit der Deklaration bennenen wir eine Variable und machen diese dem Compiler bekannt. Der Compiler ist das Programm, welches den Quellcode, also den von uns in der Programmiersprache C geschrieben Code, in ein direkt ausführbars Programm übersetzt. D.h. der Compiler weiß nun, ob wir uns im Laufe unseres Programms beim Eintippen des Variablennames vertippt haben. Da jeder Variablenname eines Datentyps eindeutig sein muss, kann der Compiler auch den Fehler abfangen, wenn versucht wird, zwei Variablen mit dem gleichen Typ und gleichen Namen zu deklarieren.

Kontrollstrukturen

Normalerweise wird Code Zeile für Zeile, von oben nach unten, ausgeführt. Manchmal möchte man aber eine Zeile - oder einen ganzen Block von Zeilen - aber nur unter einer bestimmten Bedingung durchführen. Alternativ möchte man den selben Block von Zeilen mehrfach hintereinander ausführen. Kontrollstrukturen (Steuerkonstrukte) sind Anweisungen um den Ablauf eines Computerprogramms zu steuern. Eine Kontrollstruktur ist entweder eine Verzweigung oder eine Schleife. Meist wird ihre Ausführung über logische Ausdrücke der booleschen Algebra beeinflusst.

Verzweigungen

Eine Verzweigung legt fest, welcher von zwei (oder mehr) Programmabschnitten, abhängig von einer (oder mehreren) Bedingungen, ausgeführt wird. Eine bedingte Anweisung besteht aus einer Bedingung und einem Codeabschnitt, der wiederum aus einer oder mehreren Anweisungen besteht. Wird bei der Programmausführung die bedingte Anweisung erreicht, dann wird erst die Bedingung ausgewertet, und falls diese zutrifft (und nur dann) wird anschließend der Codeabschnitt ausgeführt. Ein alltägliches Beispiel wäre dafür die Abfrage eins Passwort: Wenn(Passwort richtig) - dann: öffne das Schloss; - sonst: schlage Alarm; Danach wird in jedem Fall die Programmausführung mit den auf die bedingte Anweisung folgenden Anweisungen fortgesetzt.

Beispielcode in C:

```
int x;
x = Terminaleingabe()

//if-else-Konstrukt
if (zahl == 5)
  x = "Zahl gleich 5";
else
  x = "Zahl ungleich 5";
```

In vielen Programmiersprachen gibt es auch mehrfache Verzweigungen, auch Fallunterscheidungen genannt. Diese sind für diesen Versuch nicht vorausgesetzt, können aber bei Interesse benutzt werden. In C kann dies durch switch-Konstrukt oder if-elif-else-Konstrukt realisiert werden.

Schleifen

Eine Schleife (auch „Wiederholung“ oder englisch loop) wiederholt einen Anweisungs-Block – den sogenannten Schleifenrumpf oder Schleifenkörper –, solange die Schleifenbedingung als Laufbedingung gültig bleibt bzw. als Abbruchbedingung nicht eintritt. Schleifen, deren Schleifenbedingung immer zur Fortsetzung führt oder die keine Schleifenbedingung haben, sind Endlosschleifen. So kann man das Betriebssystem eines Computers oder Handys als Endlosschleife sehen, die immer weiter läuft, solange man nicht mit dem Befehl des Herunterfahrens die Schleife beendet. Für den Versuch werden wir Schleifen in zwei Variationen benutzen: - um den selben Code mit unterschiedlichen Werten auszuführen: Stelle die Leistung der Lampe bei jedem Durchlauf höher - um den selben Code wiederholt zu unterschiedlichen Zeiten auszuführen: Schreibe alle 5 Sekunden die aktuelle Temperatur in eine Datei. Schleifen können beliebig verschachtelt werden: Innerhalb des Schleifenkörpers der äußeren Schleife befindet sich wiederum eine Schleife, sie liegt innen, oder unter der äußeren Schleife.

Prinzipiell werden folgende Typen von Schleifen unterschieden: - kopfgesteuerte

Schleife: Bei dieser Schleife wird eine Bedingung geprüft, mit der vorher entschieden wird, ob der Schleifenrumpf (Schleifeninhalt) ausgeführt wird (meist mit WHILE = solange eingeleitet).

- Die nachprüfende oder fußgesteuerte Schleife: Bei dieser Schleife wird nach dem Durchlauf des Schleifenrumpfes (Schleifeninhalts) eine Bedingung überprüft, ob der Schleifenrumpf nochmal ausgeführt wird (meist als Konstrukt DO... WHILE = „ausführen ... solange“ oder REPEAT... UNTIL = „wiederholen ... bis“).
- Die Zählschleife, eine Sonderform der vorprüfenden Schleife (meist als FOR = für-Schleife implementiert).
- Die Mengenschleife, eine Sonderform der Zählschleife (meist als FOREACH = „für jedes Element der Menge“ implementiert; die Reihenfolge der Elemente ist beliebig).
- Schleife mit Laufbedingung: Die Schleife wird solange durchlaufen, wie die Schleifenbedingung zu „wahr“ ausgewertet wird.
- Schleife mit Abbruchbedingung: Wertet die Bedingung zu „wahr“ aus, so wird die Schleife abgebrochen.

Eine Endlosschleife ohne Schleifenbedingung (und ohne Schleifenabbruch darin) kann nur von außen unterbrochen werden, etwa durch einen Programmabbruch durch den Benutzer, Reset, Interrupt, Defekt, Abschalten des Gerätes oder ähnliches.

Für diesen Versuch ist es ausreichend die Zählschleife und die kopfgesteuerte Schleife genauer zu betrachten: - Bei einer For-Schleife zählt der Computer von einer Anfangszahl bis zu einer Endzahl und wiederholt dabei jedes Mal den Codeblock („Schleifenrumpf“). Die aktuelle Zahl wird in eine Variable („Iterator“) gesetzt, damit sie bei Bedarf in dem Codeblock Verwendung finden kann. Häufig ist die Zählschleife auf Ganzzahlen beschränkt. Das Ändern der Iterator-Variablen im Schleifenkörper ist bei vielen Programmiersprachen verboten und gilt als schlechter Programmierstil, da es oft zu schwer verständlichem Code führt – es läuft der Denkweise zuwider, direkt am Schleifenkopf die Anzahl der Durchläufe ablesen zu können.

Beispielcode in C:

```C
int i;

for(i=0; i<5; i++) {
 printf("Zahl %d\n", i+1);
}
```

- Bei einer kopfgesteuerten Schleife erfolgt die Abfrage der Bedingung, bevor der Schleifenrumpf ausgeführt wird, also am Kopf des Konstruktes. Eine

logische Operation kann beispielsweise sein: (x > 4) Solange diese Bedingung wahr ist, werden die Anweisungen innerhalb der Schleife ausgeführt. Wird der Inhalt der logischen Operation nicht im Schleifenrumpf verändert, ist diese Kontrollstruktur meist nicht die richtige, weil diese Schleife sonst kein einziges Mal durchlaufen wird oder unendlich lang läuft. Beispielcode in C:

```
int zufallszahl, rateversuch, anzahl;
zufallszahl = random_zahl();
anzahl = 0;
while (zufallszahl != rateversuch) {
  rateversuch = random_zahl();
  anzahl = anzahl + 1;
}
printf( "Die Zufallszahl lautet: %d und wurde nach %d Versuchen gefunden", rateversuch, anza
```

In diesem Beispiel wird nach der Deklaration eine Zufallszahl zufällig ausgewählt. "!=" ist der Ungleich Operator und gibt "wahre Aussage" zurück, wenn zufallszahl und rateversuch ungleich sind. Dadurch wird die Schleife so lange wiederholt, bis rateversuch gleich zufallszahl ist. Bei jedem Durchlauf wird weiterhin die Zählvariabel "anzahl" um eins erhöht. Die Bildschirausgabe erscheint erst, wenn die Schleifenbedingung nicht erfüllt wurde.

Schleifenabbruch im Sonderfall

In Fällen, die schwierig als Schleifenbedingung zu fassen sind, kann eine Schleife (aus dem Schleifenkörper heraus) meist abgebrochen werden. Mit dem Schlüsselwort "break" können wir zu jeder Zeit eine Schleife verlassen, ohne auf den Kontrollpunkt warten zu müssen. Meist gibt es einen Befehl zum Gesamt-Abbruch der Schleife, das Programm wird dann mit der ersten Anweisung nach der Schleife fortgesetzt. Beispielcode in C:

```
while (1) {
  printf("Bitte gib ein positive Zahl ein: ");
  scanf("%lf",&input);

  if(input <= 0)
  {
      printf("Nicht positiver Input: Das Programm wird beendet");
      break;
  }
  printf("Die Wurzel von %lf ist %lf \n",input, sqrt(input));
}
```

Übungsaufgaben:

In diesem Versuch werden Messwerterfassung und Prozesssteuerung durch kleine Programme verdeutlicht, die zur Versuchsdurchführung selbständig geschrieben werden. Es ist ausreichend die oben erklärten Grundstrukturen verstanden zu haben. Zwei kleine Übungsaufgaben zur Vorbereitung: Wenn Sie diese Aufgabe lösen konnt, habt Sie die nötigen Programmiervorkenntniss. Ein möglicher Online-Compiler ist (d.h. es muss keine besondere Software auf dem Rechner installiert werden): https://onlinegdb.com/

Aufgabe 1:

"Der Spieler soll eine im Programm festgelegte Zahl erraten. Dazu stehen ihm beliebig viele Versuche zur VerfÃ¼gung. Nach jedem Versuch informiert ihn das Programm darÃ¼ber, ob die geratene Zahl zu groÃŸ, zu klein oder genau richtig gewesen ist. Sobald der Spieler die Zahl erraten hat, gibt das Programm die Anzahl der Versuche aus und wird beendet" von http://python.daniel-co.de/content/praxis-zahlenraten-1.html

Aufgabe 2:

Bildschirmausgabe mit Dreieck, Raute

Erstelle ein Programm, das eine Raute auf dem Bildschirm ausgibt. Die Raute wird mittels *-Zeichen dargestellt. Die Breite der Raute ist dynamisch und kann mit einer Zahl, die eingegeben wird, bestimmt werden. Für den Anfang kann auch nur ein Dreieck ausgegeben werden. Beispiel-Ausgabe:

```
Eingabe Rauten-Breite: 5

  *
 ***
*****
 ***
  *
```

LabVIEW

LabVIEW ist ein grafisches Programmiersystem von National Instruments. Durch die Hauptanwendungsgebiete der Mess-, Regel- und Automatisierungstechnik wird dieses Werkzeug in der Industrie und in der Wissenschaft genutzt. Die Programmierung erfolgt mit einer grafischen Programmiersprache, nach dem Datenfluss-Modell.

Für einen kurzen Überblick können Sie folgendes Video gucken: https://www.youtube.com/watch?v=1umq5KqQ

Widerstandsthermometer

Für die Temperaturmessung in diesem Versuch wird ein elektrisches Bauelemente genutzt, welches den elektrischen Widerstandes in Abhängigkeit der Temperatur variiert. Als Widerstandsmaterial eignen sich vorzugsweise reine Metalle. Sie zeigen stärkere Widerstandsänderungen als Legierungen. Ferner haben sie einen nahezu linearen Zusammenhang des Widerstandes mit der Temperatur.

Herkömmliche Thermometer messen die Temperatur anhand der Längen- oder Volumenänderung eines Stoffes und sind nur als anzeigende Messgeräte geeignet. Der Vorteil der Widerstandsthermometer liegt darin, dass sie ein elektrisches Signal liefern und sich zum Einsatz in der digitalen Messtechnik eignen.

Regelungstechnik

Die prinzipielle Wirkungsweise einer Regelung kann man auch in drei Schritten verkürzt darstellen. 1.Messen: Der Istwert wird direkt gemessen oder aus anderen Messgrößen berechnet. 2.Vergleichen: Die Regelgröße wird mit dem Sollwert verglichen und die Regeldifferenzberechnet. 3.Stellen: Aus der Regeldifferenz wird die Stellgröße bestimmt.

Zweipunktregelung

Ein Zweipunktregler ist ein unstetig arbeitender Regler mit zwei Ausgangszuständen. Bei unstetigen Reglern springt die Stellgröße zwischen verschiedenen Werten. Aus diesem Grund werden unstetige Regler auch als schaltende Regler bezeichnet. Bei einer Zweipunktregelung gibt es nur zwei Möglichkeiten, das zu regelnde System zu beeinflussen. Zum Beispiel:

Heizung EIN - Heizung AUS Ventil offen - Ventil geschlossen Pumpe EIN - Pumpe AUS Linksdrehung - Rechtsdrehung Drehzahl hoch - Drehzahl niedrig

Proportionalregelung (P-Regler)

Die eben diskutierten unstetigen Regler haben den Vorteil ihrer Einfachheit. Allerdings haben unstetige Regler in der realen technischen Umsetzung auch eine Reihe von Nachteilen. Man stelle sich einen Automotor vor, dessen Drehzahl unstetig geregelt wird. Es gäbe dann nichts zwischen Leerlauf und Vollgas. Für derartige Anwendungen verwendet man daher stetige Regler. Hierbei ist die Reglerausgangsgröße proportional zur Regeldifferenz. Der Proportionalregler

ist ein proportional wirkender Regler. Das heißt die Regelabweichung und die Stellgröße stehen in einem bestimmten Verhältnis. Dieses Verhältnis wird durch den Verstärkungsfaktor Kp festgelegt. Ein Beispiel: Ein Lüfter einers Computers soll zur Kühlung der CPU geregelt werden. Der Motor des Lüftres soll sich einschalten, wenn die Temperatur um einen bestimmten Wert steigt. Ab einer Regelabweichung von 5°C soll die Motorleistung 20% betragen, bei 10°C 40%. Als Verstärkungsfaktor muss dann Kp = 4 eingestellt werden.

Integralregelung

Um die Regelung weiter zu verbessern, fügt man ggf. einen I-Anteil hinzu. Das „I“ steht für Integral. Der Integral-Anteil einer Reglung berücksichtigt nicht den “Fehler” selbst, sondern den über der Zeit aufsummierten (integrierten) Fehler. Damit führt selbst ein kleiner Fehler irgendwann zu einer Reaktion des Reglers. Das bedeutet für die Umsetzung, jedes Mal wenn der aktuelle Fehler ermittelt wird, wird er zu der Variablen (hier integral genannt) hinzuaddiert: integralteil = integralteil + fehler Am Ende wird - wie bei dem P-Anteil - der integrale Wert mit einer Konstanten multipliziert. Der I - Regler korrigiert Fehler, die sich in der VERGANGENHEIT aufsummiert haben.

Multifunktions-I/O-Gerät alias DaQ-Pad

Zur Datenerfassung und Generierung von Steuersignalen wird ein Data Acqusition Modul benutzt. Es ist per USB an den PC angeschlossene und wird mit LabVIEW programmiert. Es stellt eine direkte Schnittstelle zum Messen und Stellen von Analog- und Digitalsignal am PC dar. Es erlaubt Livemessung von nahezu beliebiger Messsignale. Dabei werden analoge Spannungssignale als Eingang von ± 10 V erwartet. Dieses Signal kann in einer Rate von 10 kHz und einer Auflösung von 14 Bit (d.h. ca. 0,6 mV) gemessen werden. Die Analoge Ausgänge werden in diesem Versuch Ansteuerung weiterer Geräte genutzt.