F15 - Manual: Input measuring and process control

Sven Brieden 22.10.2018

Introduction

In the course of the experiment, you will experience some processes for taking measurements, as well as learning about and developing strategies to control external parameters.

Such processes are often associated with many individual steps (calibrations, referencing, etc.) that must first be performed cleanly before reliable results can be generated.

The aim of today's experiment will be to show you such processes using various examples and to generate reliable data from unknown signals step by step. The sequence and sequence of the individual parts of the experiment are based on what actually occurs in the preparation of a complex research experiment.

Contents

Classification of the experiment

In order to get to know the work in the laboratory, all measuring instruments were generally controlled manually in the Grundpraktikum. Also all measurement results were recorded manually. As a result of this greatly reduced complexity of the experiments, the focus can be directed to fundamentals: evaluation of the measurement data, as well as their critical classification and the error analysis. The practice in the Fortgeschrittenenpraktikum consists of partial automation and detailed parameter scans. It is usually assumed that the measured value acquisition and process control functions without problems. We try to open this "black box" in this attempt.

Measured value acquisition

The individual processing steps of the measured value acquisition are: 1. physical process to be analyzed 2. Sensor registers a physical quantity and often provides a voltage signal. This signal is in the simplest case analog (on / off), for example, from a photoelectric sensor. Many temperature sensors and pressure sensors have a linear relationship between the measurand and the output signal. Both the type of output signal and the relationship associated with the measured variable are variably adjusted to the current problem. 3. Signal processing, improves the signal by recordable noise removal. It is optimized for the following step of analog-to-digital conversion. 4. Analog-to-digital converter converts the processed output signal of the sensor into a signal that is understandable for the PC and the standard processing paths. 5. Digital processing is usually the last step in collecting, displaying and evaluating the signal. The signals can be traced back to the measured variables and compared by means of measured value evaluation.

Programming

Data types

In the following the elementary data types are presented, which we need in the programming, in particular in the work with variables. Data types are used for better order in a program and enable effective use of the memory space.

When choosing the right data type, the following considerations play an important role: - What do I want to save or process? - Characters - Text - Numbers - Numbers with decimal points

Data types have different uses. Thus, the data type may be integer, e.g. do not store letters and numbers, only integers. Data types have different value ranges. The value range indicates which numbers this data type can store. Depending on whether you need to store small numbers up to 65535, slightly larger ones up to 4.294.967.295 or very large numbers with 20 digits, the data type will be selected accordingly.

Integer numbers: int

With the keyword int, we create variables of ordinary size. This variant is also used the most. By default, this data type is at least 16 bits, but 32 bits for a 32-bit processor. This results in a value range of -2,147,483,647 to +2,147,483,647, in the absence of a sign from 0 to 4,294,967,295.

Kommazahlen

As with the integers, the comma numbers, also known as floating-point numbers, have different variants - depending on the required size: - float with 32 Bit - double with 64 Bit - long double with 80 Bit

The exact limits of the value ranges vary from system to system. For this experiment, the precision and the range of values of a float as a number of digits is sufficient. The value range can be written by the largest number of 3 * 10 ^ 38 and an accuracy of up to 7 decimal places.

Variable declaration

```
// Variables being declared
int number, another_integer;
float small_number;
...
```

With the declaration, we name a variable and make it known to the compiler. The compiler is the program that translates the source code, ie the code written by us in the programming language C, into a directly executable program. Ie the compiler now knows if we made a typing error in the course of our program when typing in the variable name. Because each variable name of a data type must be unique, the compiler can also catch the error when trying to declare two variables of the same type and name.

Control structures

Normally, code is executed line by line, from top to bottom. Sometimes you want to do one line - or a whole block of lines - but only under a certain condition. Alternatively, one would like to execute the same block of lines several times in succession. Control structures (control constructs) are instructions to control the flow of a computer program. A control structure is either a branch or a loop. Most of their execution is influenced by logical expressions of Boolean algebra.

Branching

A branch determines which of two (or more) program sections will be executed, depending on one (or more) conditions. A conditional statement consists of a condition and a section of code, which in turn consists of one or more statements.

If the conditional statement is reached during program execution, then the condition is first evaluated, and if this is true (and only then) then the code section is executed. An everyday example would be the query one password: if(password_is_correct) - then: open(castle); - else: close(gates); Thereafter, in any case, the program execution continues with the instructions following the conditional instruction.

Example in C:

```c
int x;
x = terminalInput()

//if-else structure
if (number == 42)
  x = "Number equals 42";
else
  x = "Number is different than 42";
```

In many programming languages there are also multiple branches, also called cases or case statements. These are not required for this experiment, but can be used if interested. In C, this can be done by switch construct or if-elif-else construct.

Loops

A loop repeats an instruction block - the so-called loop body - as long as the loop condition remains valid as a running condition or a termination condition does not occur. Loops whose loop condition always leads to continuation or which have no loop condition are endless loops. So you can see the operating system of a computer or mobile phone as an infinite loop that keeps going, as long as you do not end the loop with the shutdown command. For the experiment we will use loops in two variations: - to execute the same code with different values: increase the power of the lamp with each pass - to run the same code repeatedly at different times: Write the current temperature to a file every 5 seconds. Loops can be nested arbitrarily: within the loop body of the outer loop is again a loop, it lies inside, or under the outer loop.

In principle, the following types of loops are distinguished: - head-driven loop: This loop checks a condition that determines in advance whether the loop body (loop contents) will be executed (usually with WHILE = initiated).

- The checking or foot-controlled loop: This loop checks the condition that the loop body is executed again (usually as a construct DO … WHILE = “execute … as long as” or REPEAT … UNTIL = “repeat … until after execution of the loop body (loop contents)”).
- The counting loop, a special form of the pre-checking loop (usually implemented as FOR = for-loop).

- The quantity loop, a special form of the counting loop (usually implemented as FOREACH = “for each element of the set”; the order of the elements is arbitrary).
- Loop with running condition: The loop is run as long as the loop condition is evaluated as “true”.
- Loop with abort condition: If the condition evaluates to “true”, the loop is aborted.

An infinite loop without loop condition (and without loop break in it) can only be interrupted from the outside, such as by a program abort by the user, reset, interrupt, defect, shutdown of the device or the like.

For this experiment it is sufficient to take a closer look at the counting loop and the head-controlled loop: - For a For loop, the computer counts from an initial number to a final number, repeating the code block (“loop body”) each time. The current number is placed in a variable (“iterator”) so that it can be used in the code block as needed. Often the counting loop is limited to integers. Altering the iterator variables in the loop body is forbidden in many programming languages and is considered a poor programming style, as it often leads to too difficult to understand code - it is contrary to the mindset to be able to read directly at the loop head the number of passes.

~~~~
Example in C:
``` C
int i;

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

- In a head-driven loop, the condition is queried before the loop body is executed, that is, at the head of the construct. A logical operation may be, for example: (x > 4). As long as this condition is true, the instructions within the loop are executed. If the content of the logical operation is not changed in the body of the loop, then this control structure is usually not the right one, because otherwise this loop will not be traversed once or run indefinitely. Example in C:

```c
int random_number, guess, guess_count;
random_number = random_generator();
anzahl = 0;
while (random_number != guess_count) {
  guess = random_generator();
  guess_count = guess_count + 1;
}
printf( "The random number is: %d and was found after %d attempts", guess, guess_count);
```

In this example, a random number is randomly selected after the declaration. “! =” is the unequal operator and returns “true statement” if random_number and guess are unequal. This repeats the loop until guess and random_number are equal. Each pass continues to increment the count variable guess_count by one. The screen output will not appear until the loop condition is met.

Loop termination in special case

In cases that are difficult to grasp as a loop condition, a loop (out of the loop body) can usually be broken off. With the keyword “break” we can leave a loop at any time without having to wait for the checkpoint. Mostly there is a command to abort the loop altogether, the program then continues with the first statement after the loop. Example code in C:

```
while (1) {
  printf("Please give a positive number: ");
  scanf("%lf",&input);

  if(input <= 0)
  {
      printf("The given number is not positive: Ciao");
      break;
  }
  printf("The square root of %lf is %lf \n",input, sqrt(input));
}
```

Exercises:

In this experiment, measured value input and process control are illustrated by small programs that are written independently to carry out the experiment. It is sufficient to have understood the basic structures explained above. Two small exercises to prepare: If you can solve this task, you have the necessary programming skills. A possible online compiler is (that is, no special software needs to be installed on the machine): https://onlinegdb.com/

Exercise 1:

“The player should guess a number set in the program, and any number of attempts will be available to him. After each attempt, the program informs them whether the guess was too big, too small, or just right as soon as the player guesses the number the program returns the number of attempts and ends” von http://python.daniel-co.de/content/praxis-zahlenraten-1.html

Aufgabe 2:

Screen output with triangle, diamond.

Create a program that displays a diamond on the screen. The diamond is represented with * signs. The width of the diamond is dynamic and can be determined with a number that is entered. For starters, only one triangle can be output. Sample output:

```
Input diamond width: 5

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

LabVIEW

LabVIEW is a National Instruments graphical programming system. Due to the main applications of measuring, control and automation technology, this tool is used in industry and science. The programming is done with a graphical programming language, according to the data flow model.

For a brief overview you can watch the following video: https://www.youtube.com/watch?v=1umq5KqQWMo

Resistance thermometer

For the temperature measurement in this experiment, an electrical component is used which varies the electrical resistance as a function of the temperature. Resistance materials are preferably pure metals since they show stronger resistance changes than alloys. Furthermore, they have a nearly linear relationship of resistance to temperature.

Conventional thermometers measure the temperature based on the change in length or volume of a substance and are only suitable as indicating measuring devices. The advantage of resistance thermometers is that they provide an electrical signal and are suitable for use in digital measurement technology.

Control techniques

The principle of operation of a control can be summarized in three steps: 1.Measuring: The actual value is measured directly or calculated from other measured quantities. 2. Comparison: The controlled variable is compared with the setpoint and the control difference is calculated. 3.Adjustment: The manipulated variable is determined from the control difference.

Two point control

A two-position controller is an unsteadily operating controller with two output states. With discontinuous controllers, the manipulated variable jumps between different values. For this reason discontinuous controllers are also referred to as switching controllers. There are only two ways of influencing the system to be controlled in a two-point control. For example:

Heating ON - Heating OFF Vent open - Vent closed Pump ON - Pump OFF Turn left - Turn right High speed - Low speed

Proportional control (P-controller)

The discontinuous controllers just discussed have the advantage of their simplicity. However, discontinuous controllers in the real technical implementation also have a number of disadvantages. Imagine a car engine whose speed is discontinuously regulated. There would be nothing between idle and full throttle. Therefore, continuous controllers are used for such applications. The controller output is proportional to the control difference. The proportional controller is a proportional controller. This means that the control deviation and the manipulated variable are in a certain ratio. This ratio is determined by the gain Kp. An example: A fan of a computer should be controlled to cool the CPU. The engine of the fan should turn on when the temperature rises by a certain amount. From a control deviation of 5°C the engine power should be 20%, at 10°C 40%. Then the gain factor must be set to Kp = 4.

Integral control

To further improve the regulation, you may add an I component. The "I" stands for integral. The integral part of a regulation does not take account of the "error" itself, but of the (integrated) error accumulated over time. Thus, even a small error eventually leads to a reaction of the controller. This means for the implementation, every time the current error is detected, it is added to the variable (here called integral): integral part = integral part + error At the end, as with the P component, the integral value is multiplied by a constant. The I-controller corrects errors that have accumulated in the PAST.

Multifunction I/O device alias DaQ-Pad

For data acquisition and generation of control signals a Data Acqusition module is used. It is connected to the PC via USB and is programmed with LabVIEW. It provides a direct interface for measuring and setting analogue and digital signals on the PC. It allows live measurement of almost any measuring signals. Analog voltage signals are expected as input of $\pm$ 10V. This signal can be measured at a rate of 10 kHz and a resolution of 14 bits (i.e., about 0.6 mV). The analog outputs are used in this experiment control of other devices.