COMPUTERGESTÜTZTES EXPERIMENTIEREN II
(Computer-Assisted Experimentation II)

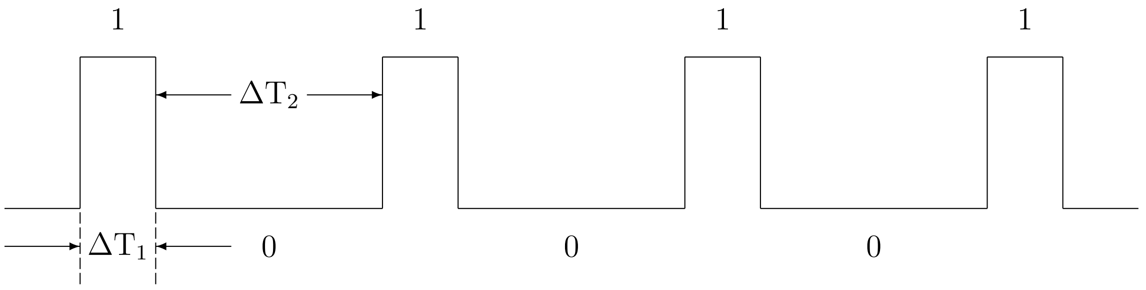
P R A C T I C A L
Programming in C/C++

Aperture Experiment

In the Aperture Experiment a light beam is imaged onto a slit aperture behind which a light detector measures the intensity of the incoming light. The slit aperture can be opened and closed using a stepping motor. In the first part of the exercise you will learn to control the stepping motor, in the second part you will learn to aquire the signal using an analog/digital converter (ADC) that measures the light intensity, and in the third part you will implement a feedback control that keeps a given light intensity shining on the detector.

1. Controlling a stepping motor

The stepping motor will be controlled using the digital output of a NI USB-6009 Interface. Bit0 and bit1 of port0 will be used. Bit0 controls the rotation (1 = clockwise, 0 = couterclockwise, i.e. opening the aperture slit). Bit1 generates impulses that trigger a step of the motor. Setting and resetting bit1 creates these impulses:



For impulses we have $\Delta T_1 \ll \Delta T_2$. The duration of a pulse ΔT_1 my be rather short, e.g. a few μs. The time the computer needs for setting and resetting the bit is sufficient. The time between pulses ΔT_2 should be in the order of 100 ms and may be generated using the procedure void Sleep(unsigned long ms).

The slit aperture is moved, - opened or closed-, using the stepping motor. There are stops on the aperture when it is completely opened or closed. Two switches sense when the aperture reaches a stop. The switches are connetcted to port1 (configured for input). Bit0 = 1 signals when the aperture is completely opened and bit1 = 1 indicates the completely closed aperture.

For the operation of the stepping motor the code of the following functionality should be implemented, as functions or procedures:

configPorts()	configure port0 for output and port1 for input as done in the previous practical exercize
n = SwitchState()	read the switches and return an integer value n of the two lowest bits of port1
openStep()	the motor does one step in the open direction (counterclockwise). Before the step is executed the SwitchState is checked and only if it is allowed the step is executed, i.e. only if the aperture is not completely open
closeStep()	the motor does one step in the close direction (clockwise). Before the step is executed the SwitchState is checked and only if it is allowed the step is executed, i.e. only if the aperture is not completely closed

Create a program in a VCL Form application that uses the code specified above. Use LabelledEdit windows for input of the steps to be done and for output of the actually executed steps, and for the relative postion given in steps. The functionality should be controlled by buttons:

Close	close the aperture, i.e. move the stepping motor clockwise by the given number of steps
Open	open the aperture, i.e. move the stepping motor counterclockwise by the given number of steps
Shut	shut the aperture, i.e. close completely. Starting from this position the absolute position of the aperture can be recorded in terms of steps by resetting the relative position to zero.

Objectives: control of a stepping motor, waiting times of different magnitude, factorization of an automization task into elementary control functions.

COMPUTERGESTÜTZTES EXPERIMENTIEREN II
(Computer-Assisted Experimentation II)

P R A C T I C A L
Programming in C/C++

Aperture Experiment
2. Data aquisition using an ADC

The intensity of the light passing through the aperture is measured using a light detector, which is a simple photo resistor. The detector creates a signal between 0 V and 5 V, that is proportional to the intensity. The signal voltage is measured with the analog/digital converter (ADC) provided by the NI USB-6009 Interface. Details of its functionality may be found in the separate document Appendix NIDAQmx.

Implement the code that sets up the ADC and that performs the A/D conversion. The former may be encapsulated in a procedure configADC(). The signal of the light detector is connected to channel0 of the multiplexer. The set up is done in the procedure CreateAIVoltageChan: choose the analog input channel ai0 and differential measurement in voltage range ± 10 V. The A/D concersion should be implemented as function:

volt = ADC() volt: the value of the measured voltage

Add a button and a LabelledEdit window that measures the voltage and show the value in the window.

Add another button that measures the curve of the signal voltage as a function of the aperture's width given in steps of the motor

voltage = f(steps)

and that displays the curve in a graph using TChart. It is recommended to proceed in the following way: first, close the aperture completely, second, open the aperture for 400 steps (as there is no signal due to the overlapping blades of the aperture) and, third, measure the curve over 1500 steps.

Objectives: aquisition and processing of analog signals, low level programming of interfaces. Presentation of graphs in TChart.

COMPUTERGESTÜTZTES EXPERIMENTIEREN II
(Computer-Assisted Experimentation II)

P R A C T I C A L
Programming in C/C++

Aperture Experiment
3. Feedback control

In this exercise you will write a program that controls the aperture such that a given intensity of the light on the detector remains constant (closed loop feedback control).

If, for illustration, the intensity of the beam is reduced by partially blocking the light source, the aperture has to be opened until the preset intensity is reached.

Based on the program of the previous exercise add an option, that first records and displays the "response curve". Then the desired value shall be given e.g. by using a graphic cursor. While the system performs the feedback control and approaches the desired value, the actual value shall be tracked in the graph.

As feedback control is implemented in an infinite loop, all other GUI functionality is blocked while it runs. As a consequence the feedback control loop can not be interrupted. A remedy would be to implement the code of the feedback control in a separate thread. This requires to enter into the complex area of multithreading. Here we just learn to create a thread (instantiate), to pause the execution of the thread (suspend), and to restart the execution (resume).

Objectives: implementation of a feedback control. Multithreading: instantiation, suspend, resume