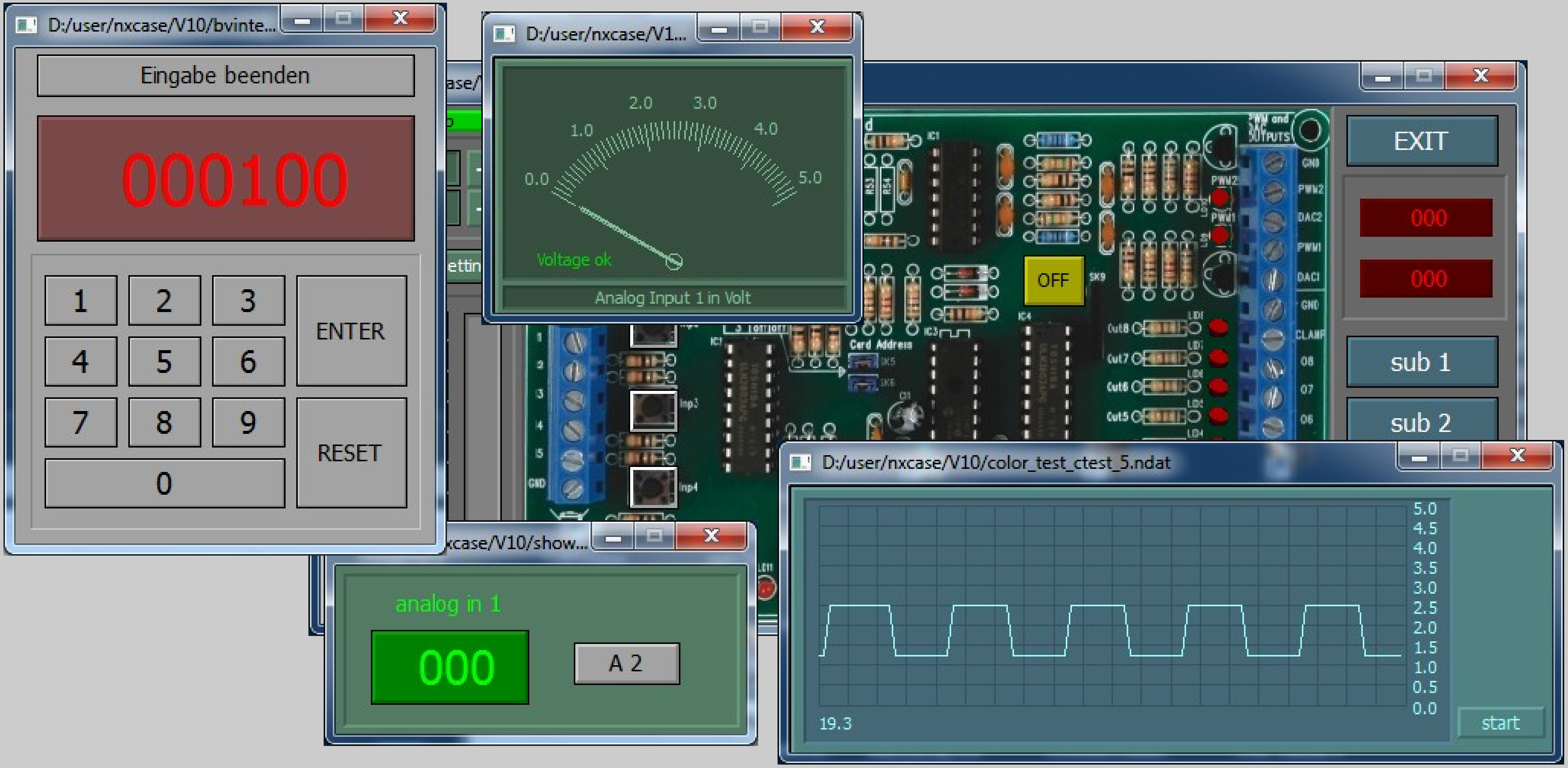
RptGui Tool


D:/user/nxcase/V10/bvinte...
Eingabe beenden
000100
1
2
3
4
5
6
7
8
9
0
ENTER
RESET
D:/user/nxcase/V1...
0.0
1.0
2.0
3.0
4.0
5.0
Voltage ok
Analog Input 1 in Volt
EXIT
000
000
sub 1
sub 2
OFF
xcase/V10/show...
analog in 1
000
A 2
D:/user/nxcase/V10/color_test_ctest_5.ndat
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
19.3
start

RptGui Tool

Rapid Proto Typing

Program for creating graphical user interfaces for control tasks without programming under Windows and Linux

RptGui Tool

- For the creation of graphical user interfaces there are generally tools based on C ++, which offer a variety of classes to enable an individual design.
- The most used are e.g. Microsoft Foundation Classes, Borland Builder and Qt class libraries.
- All of these tools provide an interface for a user program to the API of the corresponding operating system.

RptGui Tool

- The advantage of these toolkits is that you can use them to create both technical and all other graphical applications individually.
- However, if you want to create a program quickly, you will find that a good knowledge of object-oriented programming is required here, as well as a considerable effort to familiarize yourself with the corresponding class libraries.
- However, many tools are limited to one operating system.

RptGui Tool

Why RptGui?

- Most applications in the control area and the simulation of technical processes only require a limited number of graphic elements.
- Most programmers in the field of micro controller applications are only familiar with the 'C' language.

RptGui Tool

- RptGui allows the creation of graphical interfaces (technical applications) without familiarization with 'C ++' and the corresponding class libraries.
- The graphical user interface can be generated like a CAD tool and provides an interface for data exchange with programs written in 'C'.

RptGui Tool

- Arguments for RptGui

- Graphic design
- Application modularity
- Reusability of the modules
- Operating system independent
- Network capable
- Ready to run immediately

Elements in RptGui

- Windows
- Keypicks
- Objects
- Commands
- Activities
- User interface

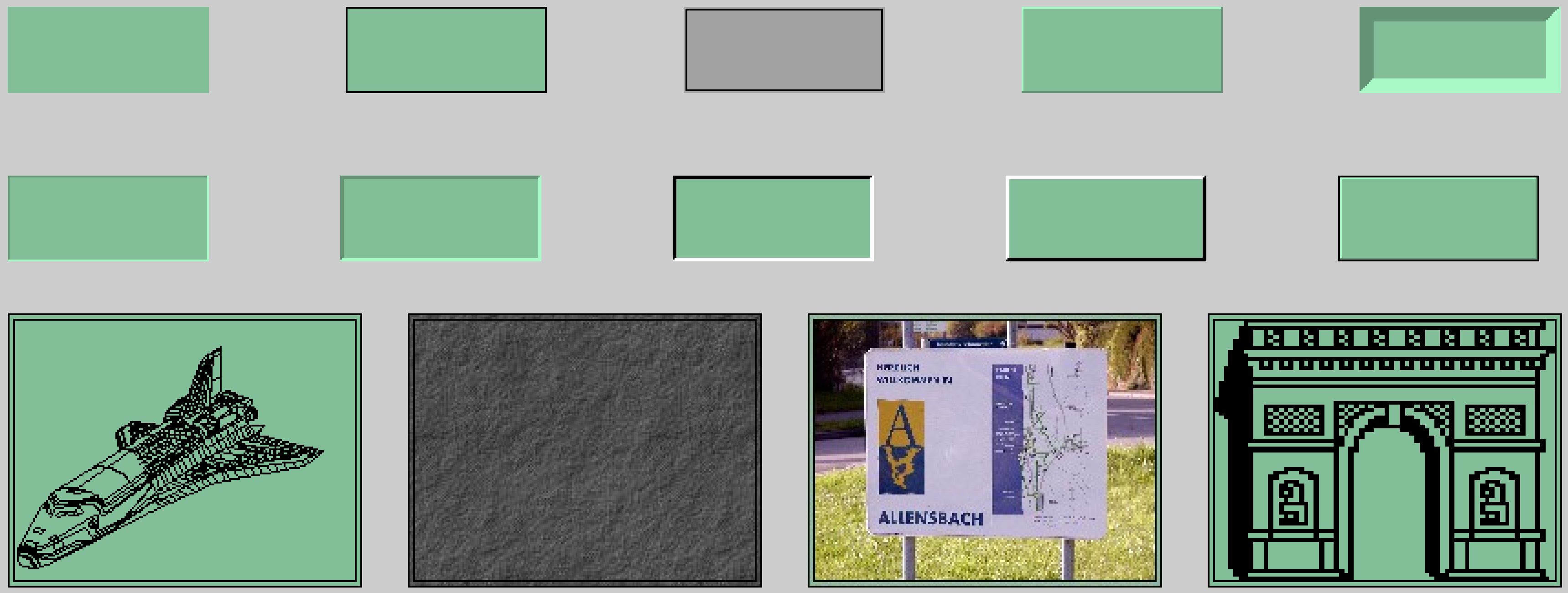
Elements in RptGui

Windows represent drawing areas, which are mainly used for design.

Windows are defined and displayed in their properties by a 'Mode'.

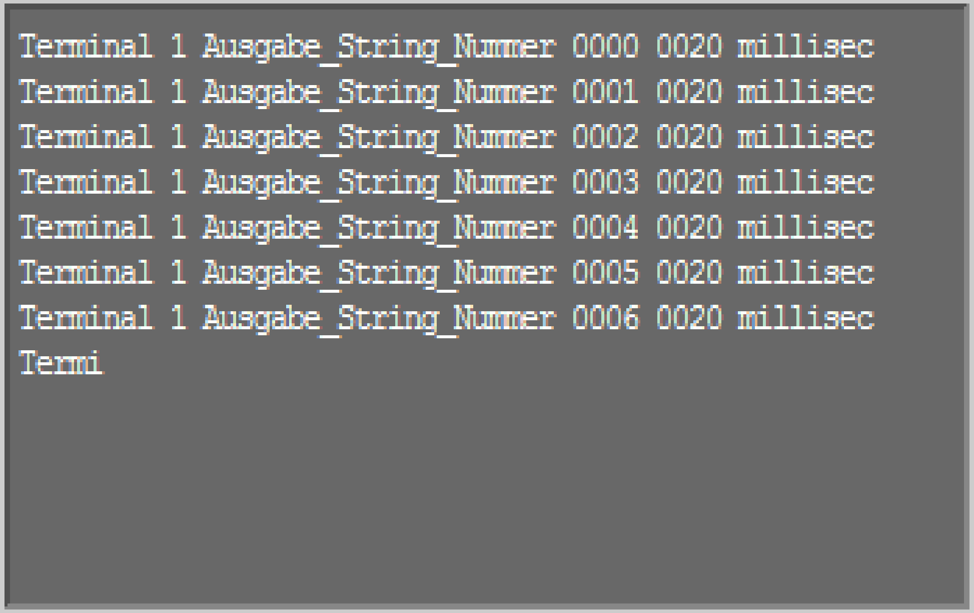
Elements in RptGui

- Examples for Windows 'Mode'



Elements in RptGui

- As a special function, a window can be configured as a terminal output.
- 10 independent terminals are possible



```
Terminal 1 Ausgabe_String_Nummer 0000 0020 millisec
Terminal 1 Ausgabe_String_Nummer 0001 0020 millisec
Terminal 1 Ausgabe_String_Nummer 0002 0020 millisec
Terminal 1 Ausgabe_String_Nummer 0003 0020 millisec
Terminal 1 Ausgabe_String_Nummer 0004 0020 millisec
Terminal 1 Ausgabe_String_Nummer 0005 0020 millisec
Terminal 1 Ausgabe_String_Nummer 0006 0020 millisec
Termi
```

- The input is made via FIFO's in the interface

Elements in RptGui

Keypicks are graphic elements which have properties similar to Windows.

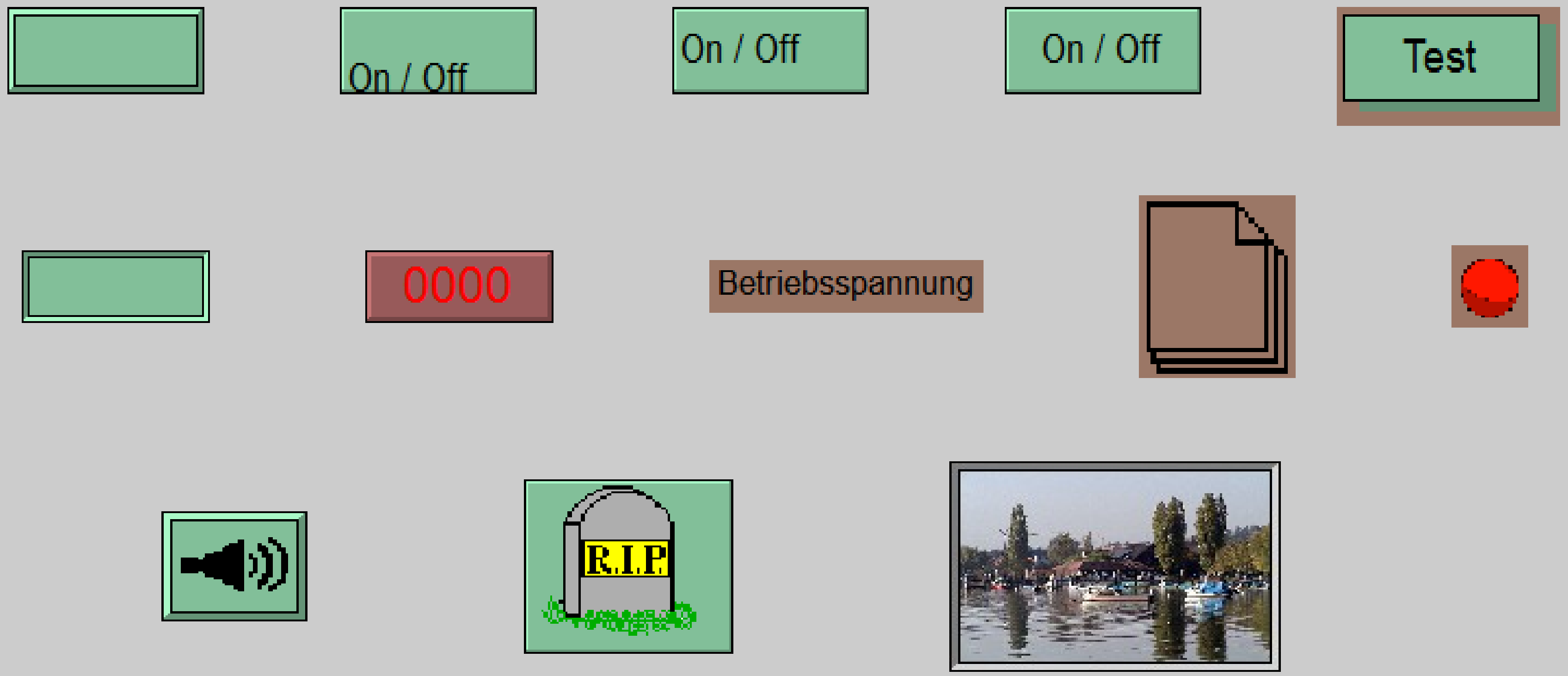
However, keypicks have other features..

An essential feature is that a keypick on a mouse click can trigger a command or change its appearance.

Each keypick has a normal and an alternative display.

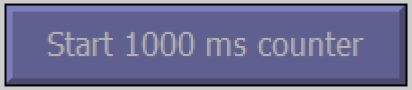
Elements in RptGui

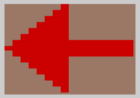
- Examples of keypicks 'mode'

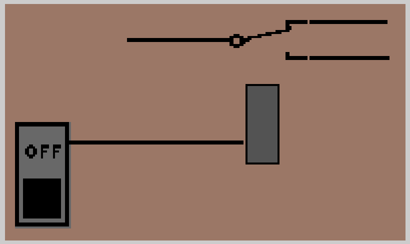


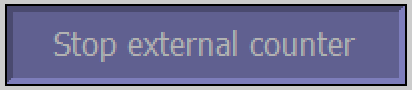
Elements in RptGui

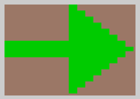
- Examples of alternative representation of a keypick

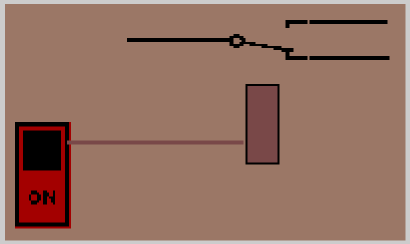










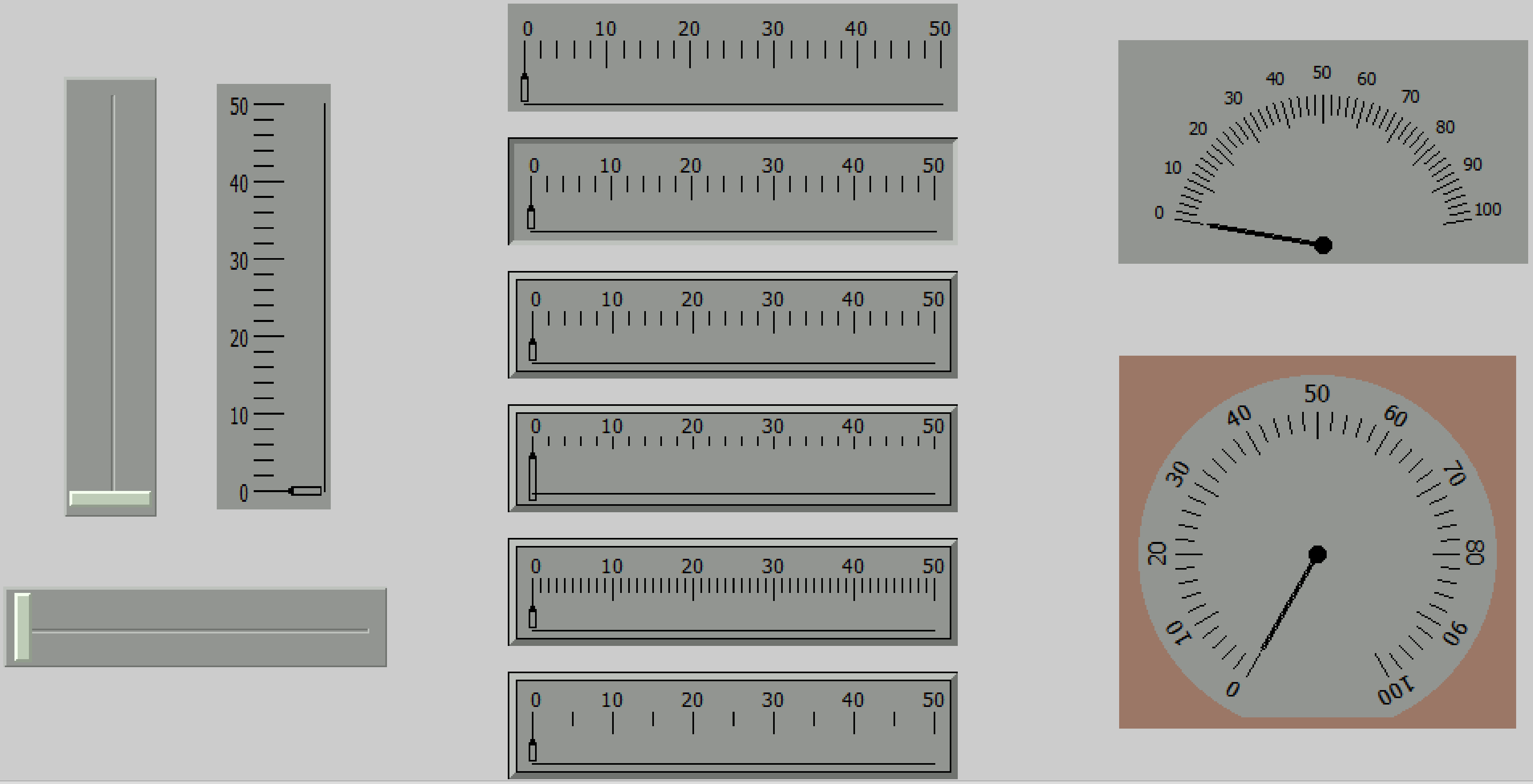


Elements in RptGui

- **Objects** represent complex elements.
    - Slider
    - Instruments
    - Chart fields

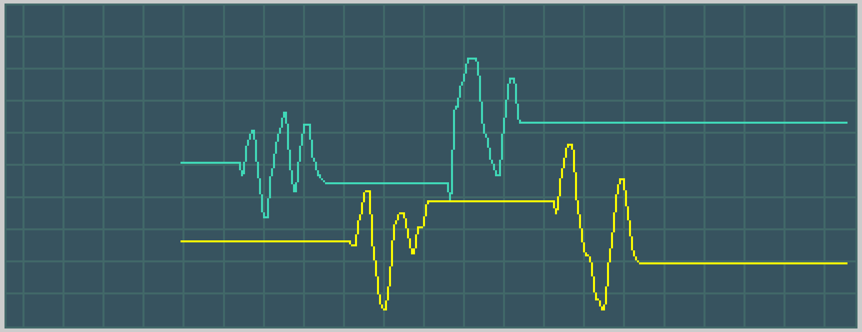
Elements in RptGui

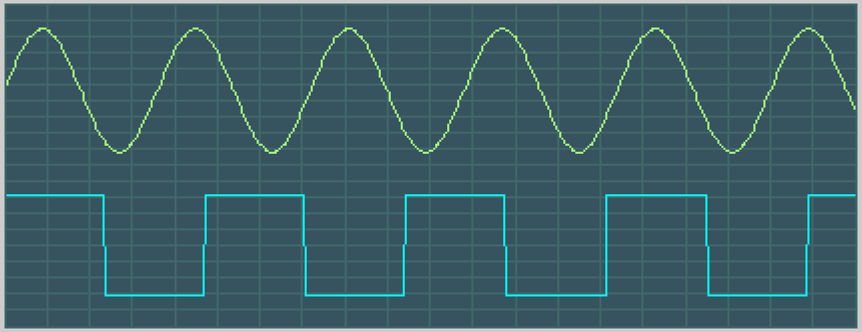
- Examples of sliders and instruments

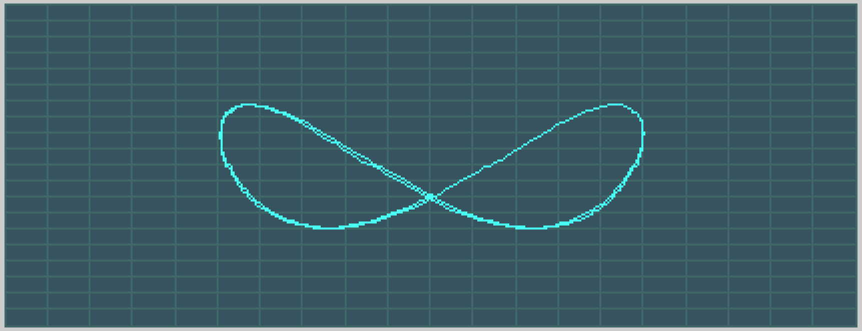


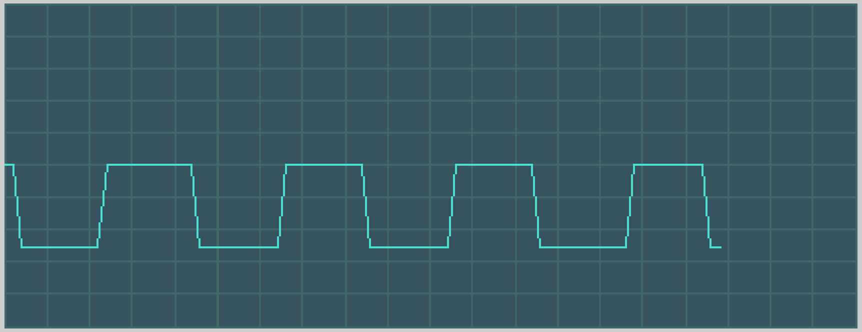
Elements in RptGui

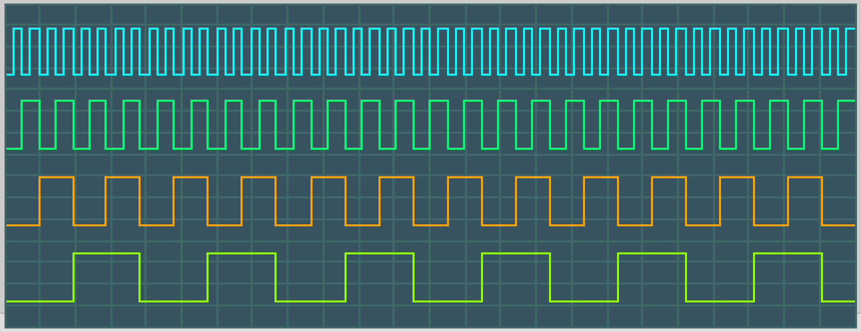
- Beispiele für Diagramme











Elements in RptGui

- **Commands** are functions that can be triggered by:
  - Mouse key down
  - Mouse key up
  - Mouse key pressed
  - Application start
  - Application end
  - Application loop.

Elements in RptGui

- **Commands** are divided into 3 categories:
  - System commands
  - Internal commands
  - External programs

Elements in RptGui

- System commands
  - chain to programm (file.ndat)
  - call submenue (file.nsub)
  - exit submenue ()
  - chain to submenue (file.nsub)
  - exit programm ()
  - start programm (file.ndat)
  - exit all programms()

Elements in RptGui

- Internal commands

- `setwert(int index, int value)`
- `setbit(int index, int bit)`
- `resbit(int index, int bit)`
- `invbit(int index, int bit)`
- `increment(int index, int value)`
- `decrement(int index, int value)`
- `increment_to_limit(int index, int value, int limit)`
- `decrement_to_limit(int index, int value, int limit)`

Elements in RptGui

- Internal commands

- `eingabe(int index, int stellen, int value)`
- `hexinput(int index, int stellen, int value)`
- `copy(int source, int destination)`
- `tofloat(int source, float dest, int faktor)`
- `scale(int source, int dest, int percent)`
- `shiftleft(int source, int dest, int pos, int mask)`
- `shiftright(int source, int dest, int pos, int mask)`
- `setmaskedwert(int index, int value , int mask)`
- `copymasked(int source, int destination , int mask)`

Elements in RptGui

- External programs
- Are called like console programs with parameter transfer.
- A library is available for connection to the RPT interface.
- An external program can be one-time.
- Or run as a parallel task in a time loop.

Elements in RptGui

- **Actions** are reactions of keypicks to the status of the program interface.
- An action forces the alternative representation of a keypick.

Elements in RptGui

The triggering can take place through the following events in the interface:

- Bit set
- Value lower than default
- Value equal to the default
- Value greater than default

Elements in RptGui

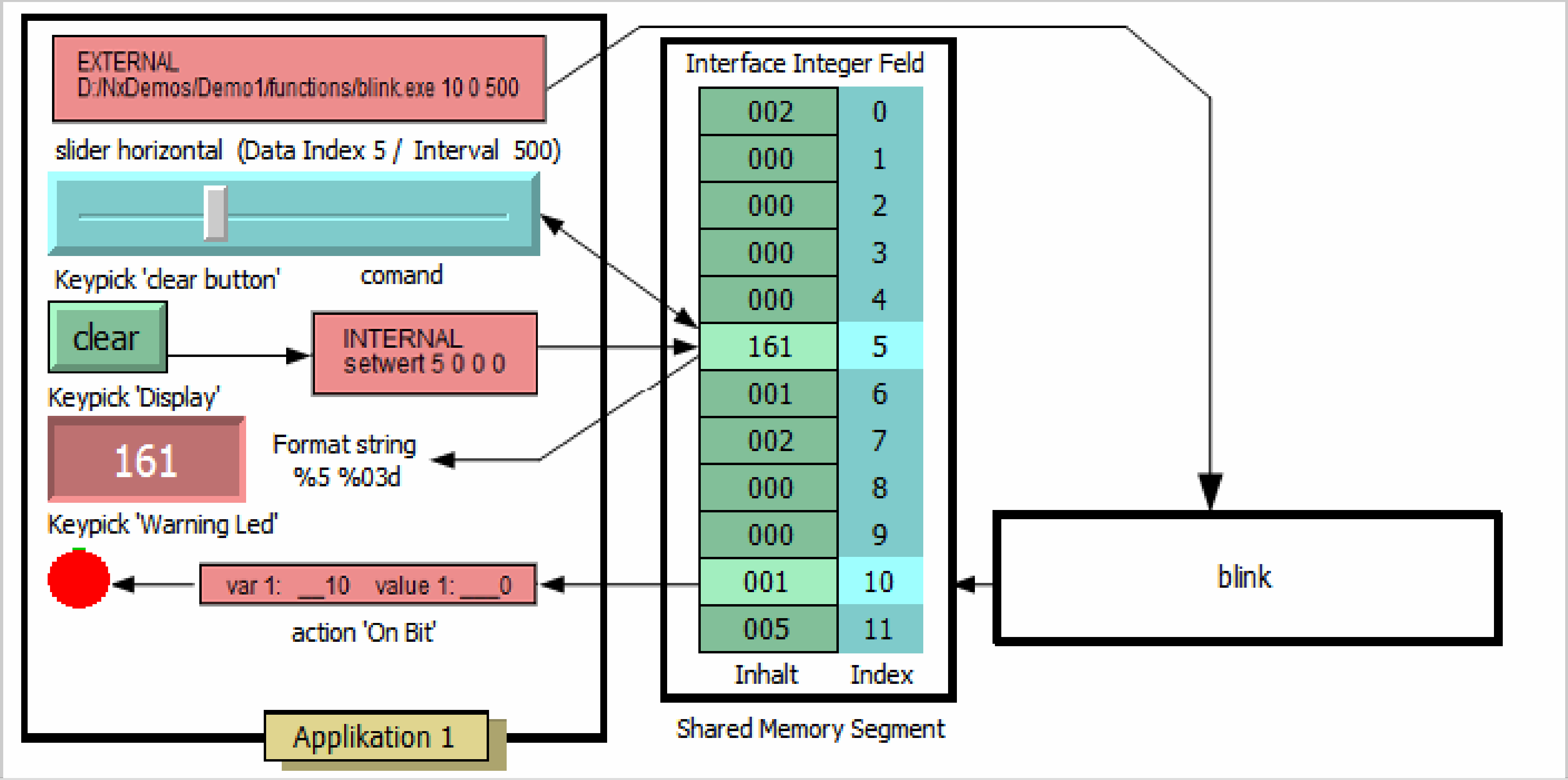
- In addition, the following properties of a keypick can be changed using actions:
  - Keypick color
  - X position of the keypick
  - Y position of the keypick
  - Keypick width
  - Keypick height.
  - Rotation of the keypick

Elements in RptGui

- RPT Interface
- The interface of the program is formed by a shared memory segment for interprocess communication.
- The interface can be accessed internally from the commands and activities.
- The data exchange between the elements of an application and with the elements of several applications as well as the data exchange with user programs takes place via the interface.

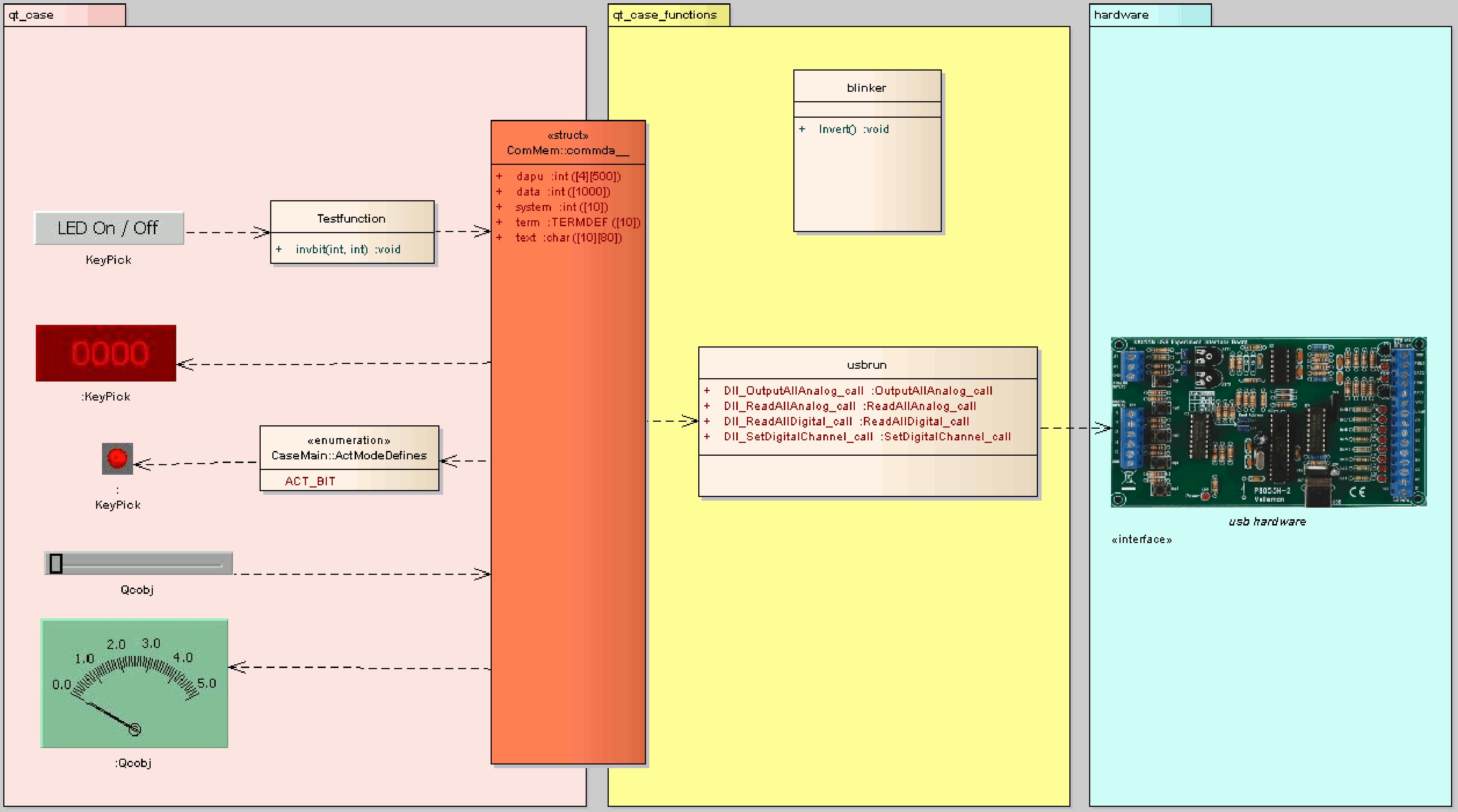
Elements in RptGui

- RPT Interface



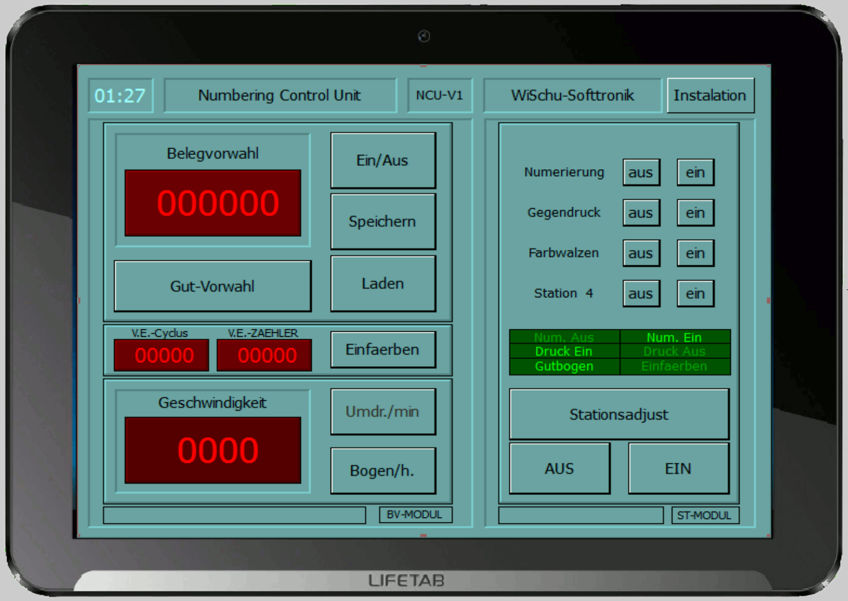
Inhalt	Index
002	0
000	1
000	2
000	3
000	4
161	5
001	6
002	7
000	8
000	9
001	10
005	11

Software structure



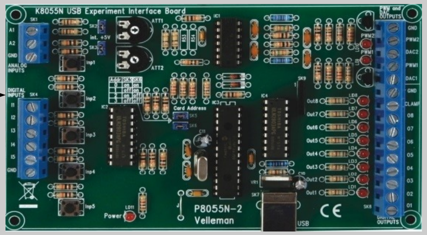
Hardware combinations

Tablet computer as a user interface



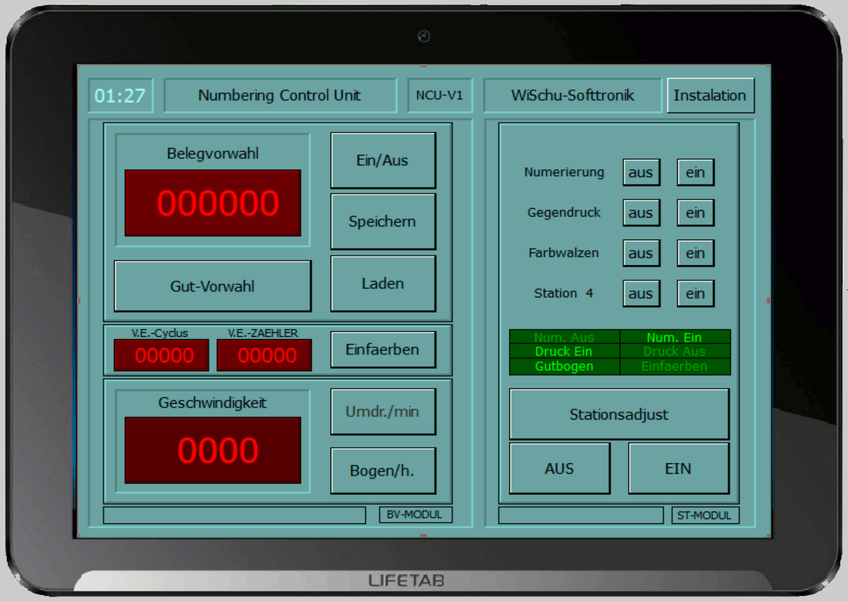
Usb board as an interface to the hardware

USB interface

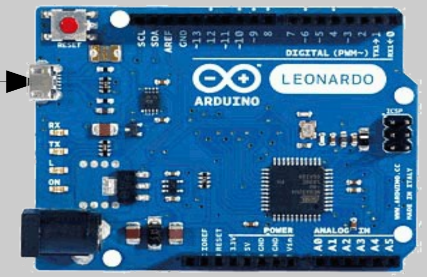


Hardware combinations

Tablet computer as a user interface

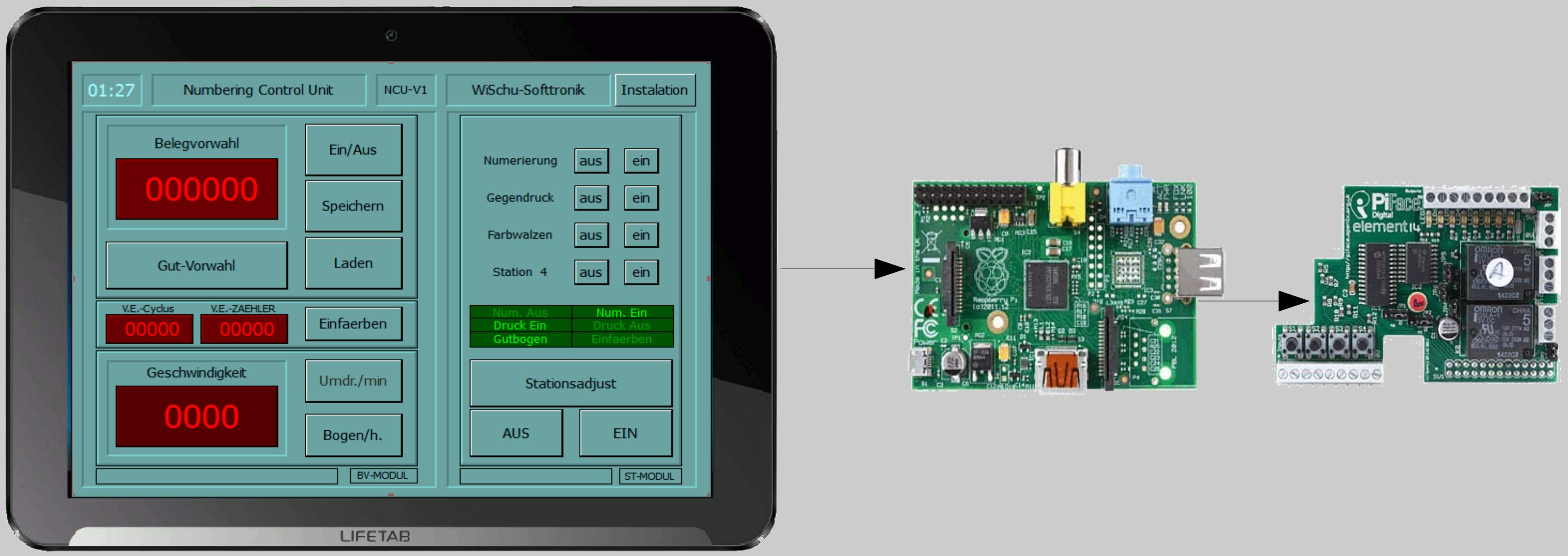


Controller with USB interface



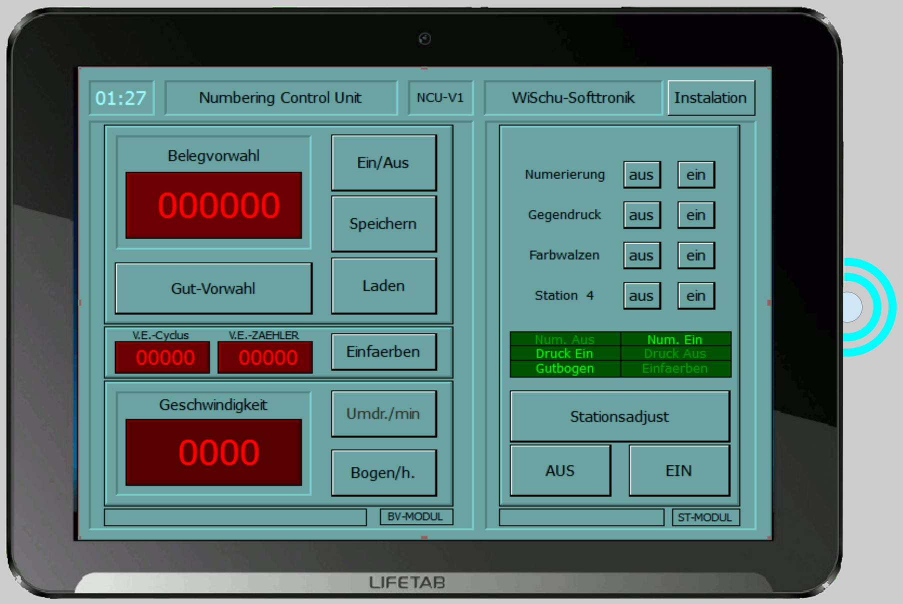
Hardware combinations

Raspberry controller with LCD display and touchscreen

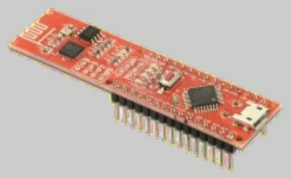


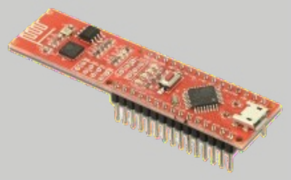
Hardware combinations

Tablet computer as an operator interface



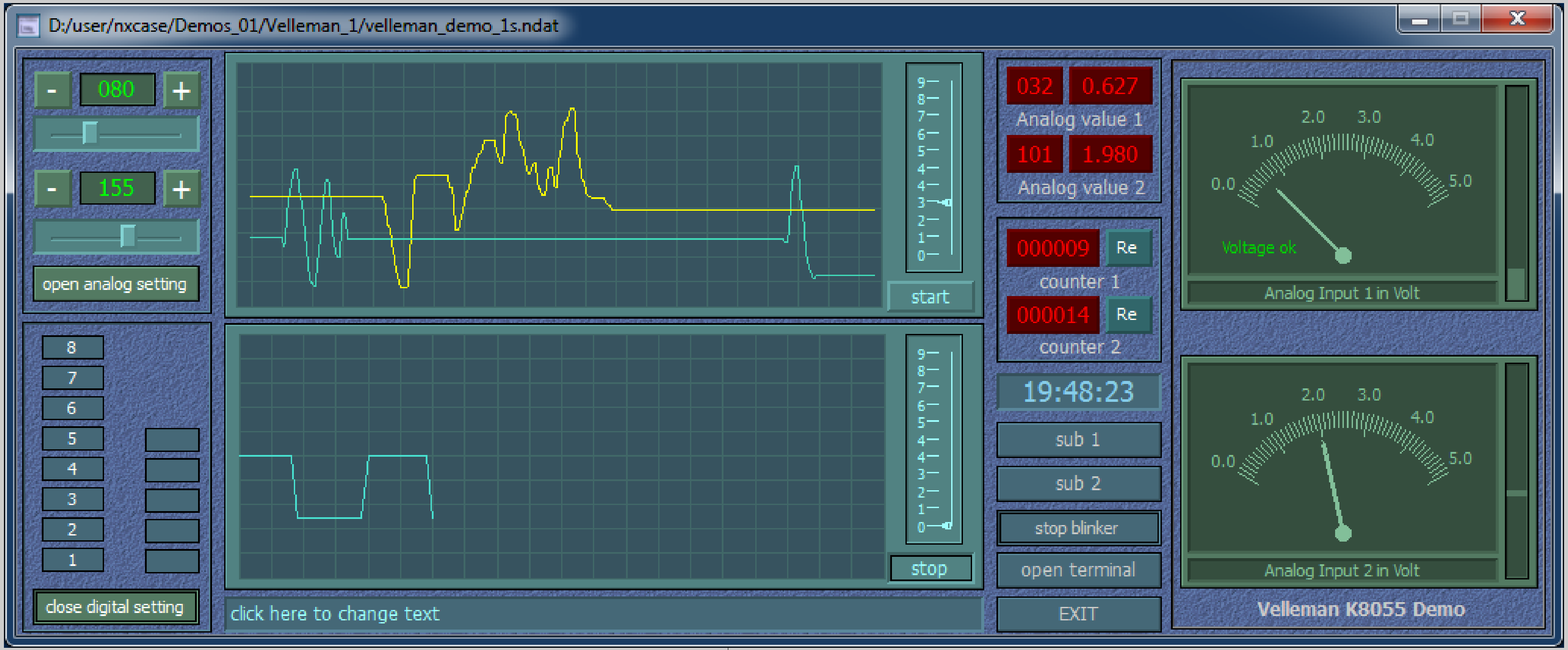
Pretzel Board - IoT WiFi Board

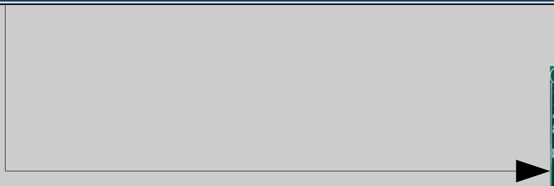


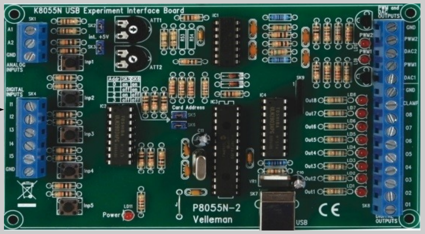


Examples

- Demo for Velleman USB IO board K8055

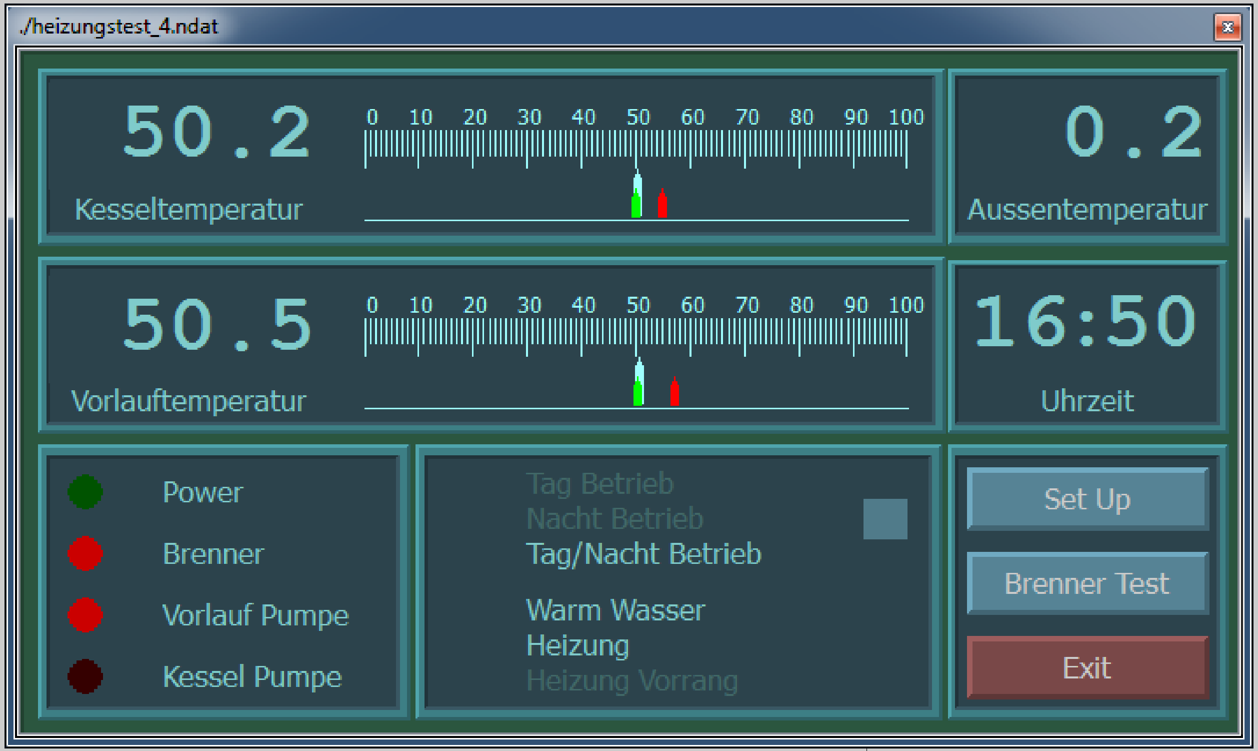




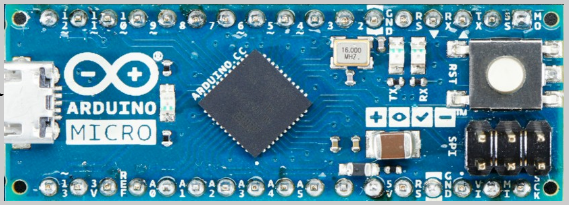


Examples

- Heizungssteuerung mit Arduio Micro



USB Verbindung



Example user program settext

```
#include <stdlib.h>
#include "../fx_common/include/fxinterface.h"

int main(int argc, char *argv[])
{
   int index = 0;
   if (argc >= 3)
   {
      if(rpt_interface_exists("wtymem"))
      {
         index = atoi(argv[1]); // index
         set_text(index,argv[2]) ;
      }
   }
}

void user_init(void)
{
}

void user_call(void)
{
 /* not  used in single call */
}

void end_call(void)
{
}
```

Example user program flashing

```
#include <stdlib.h>
#include "../fx_common/include/fxinterface.h"

int n, m, v, cvar ;

int main(int argc, char **argv)
{ if (argc >= 4)
   { if (rpt_interface_exists("wtymem"))
      { ProNum = 0;
         n = atoi(argv[1]); // index
         v = atoi(argv[2]); // bitnumber
         m = atoi(argv[3]); // time

         if (argc >= 4) { ProNum = atoi(argv[4]); }  // process number
         if (prog_controll(ProNum)) { start_loop(argc, argv, ProNum, m); }
      }
   }
   return 0;
}

void user_init(void)  { }

void user_call(void)
{
   /* *********************** include user action here **************** */
   cvar = get_var(n) ;
   if (cvar & (0x01 << v)) { cvar &= ~(0x01 << v); }
   else { cvar |= (0x01 << v); }
   set_var(n,cvar) ;
   /* ***********************   end of user action     **************** */
}

void end_call(void) { }
```