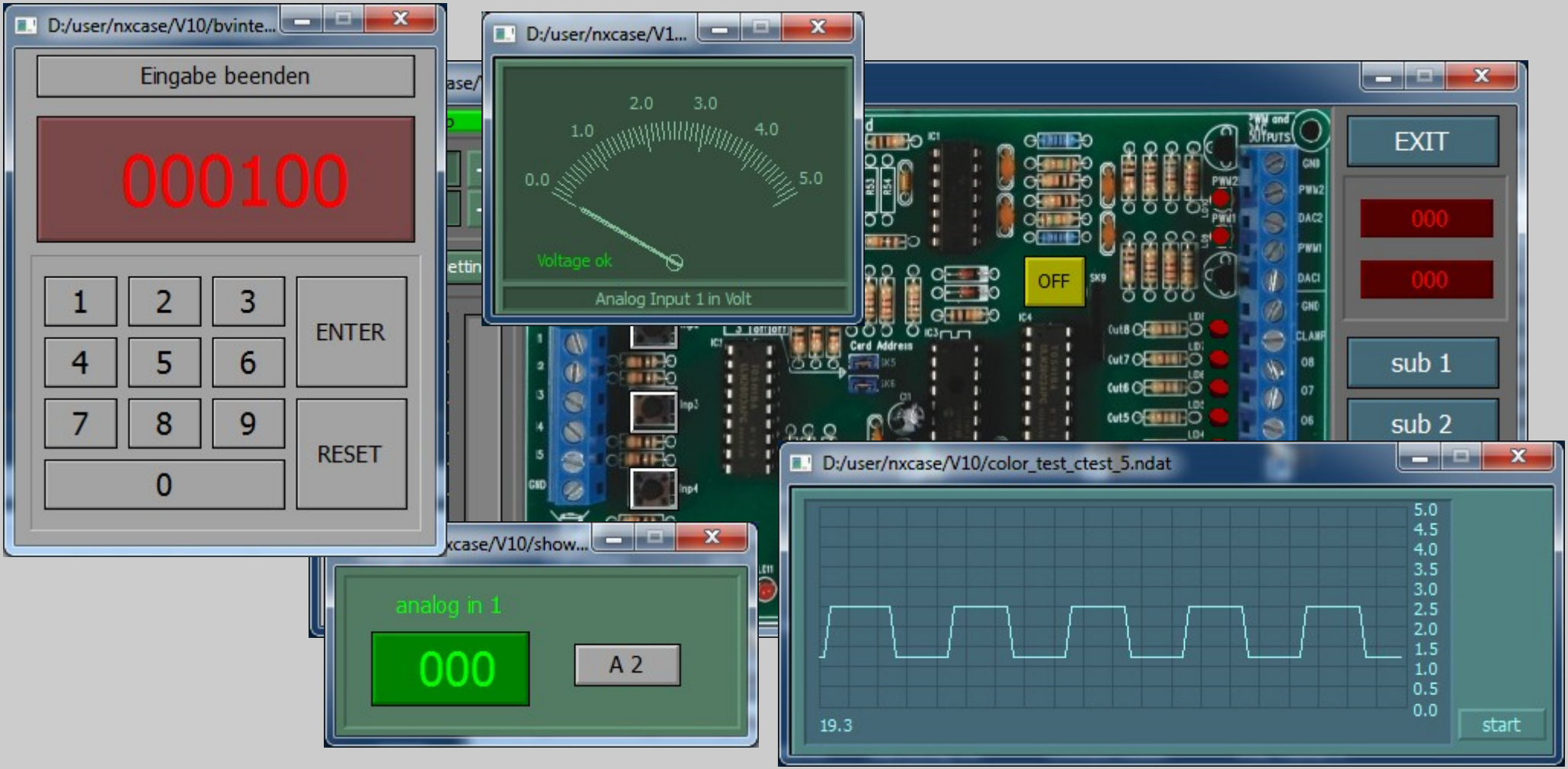
RPT_One GUI Tool



Rapid Proto Typing

Erstellung Graphischer User Interfaces für Steuerungsaufgaben ohne Programmierung unter Windows und Linux.

RPT_One GUI Tool

- Für die Erstellung graphischer User Interfaces gibt es im allgemeinen auf C++ basierende Tools welche eine Vielzahl von Klassen anbieten um ein individuelles Design zu ermöglichen.
- Die meist benutzten sind z.B. Microsoft Foundation Classes, Borland Builder und Qt Klassenbibliotheken.
- Alle diese Tools stellen ein Interface für ein Anwenderprogramm zur API des entsprechenden Betriebssystems zur Verfügung.

RPT_One GUI Tool

- Der Vorteil dieser Toolkits besteht darin, das man mit Ihnen sowohl technische als auch alle anderen graphischen Applikationen individuell erzeugen kann.
- Will man mal schnell ein Programm erstellen, stellt man jedoch fest, das hier gute Kenntnisse der Objekt orientierten Programmierung erforderlich sind, als auch ein erheblicher Aufwand an Einarbeitung in die entsprechenden Klassenbibliotheken anfällt.
- Viele Tools sind jedoch auf ein Betriebssystem beschränkt.

Warum Rpt_One ?

- Die meisten Anwendungen im Steuerungsbereich und der Simulation von technischen Abläufen benötigtigen nur eine begrenzte Anzahl von graphischen Elementen.
- Die meisten Programmierer aus dem Bereich der Micro Controller Anwendungen sind lediglich mit der Sprache 'C' vertraut.

RPT_One GUI Tool

- Rpt_One erlaubt die Erstellung graphischer Interfaces (technische Anwendungen) ohne die Einarbeitung in 'C++' und der entsprechenden Klassen Bibliotheken.
- Die graphische Bedienoberfläche kann wie mit einem CAD Tool erzeugt werden, und stellt ein Interface zum Datenaustausch mit in 'C' geschriebenen Programmen zur Verfügung.

RPT_One GUI Tool

- Argumente für Rpt-One

- Graphisches Design
- Modularität der Applikation
- Wiederverwendbarkeit der Module
- Betriebssystem unabhängig
- Netzwerkfähig
- Sofort Lauffähig

Elemente in RptOne

- Windows
- Keypicks
- Objekte
- Kommmandos
- Aktivitäten
- User Interface

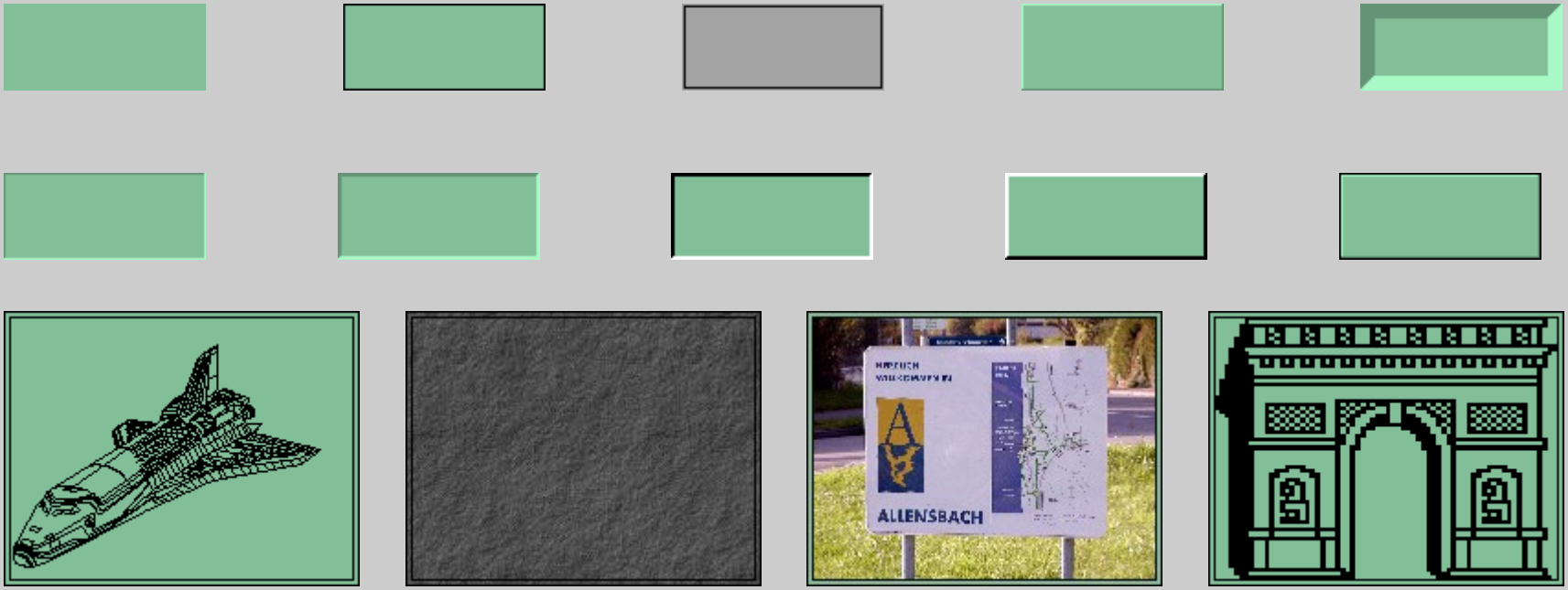
Elemente in RptOne

Windows stellen Zeichenflächen dar, welche hauptsächlich dem Design dienen.

Windows werden durch einen 'Mode' in ihren Eigenschaften definiert und dargestellt.

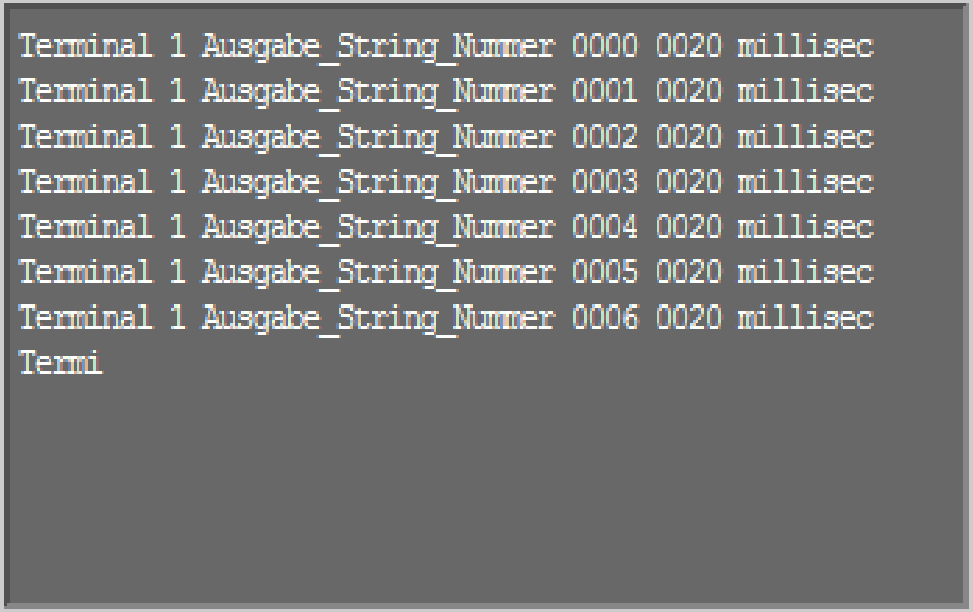
Elemente in RptOne

- Beispiele für Windows 'Mode'



Elemente in RptOne

- Als Sonderfunktion kann ein Window als Terminalausgabe konfiguriert werden.
- Es sind 10 unabhängige Terminals möglich



- Die Eingabe erfolgt über FIFO's im Interface

Elemente in RptOne

Keypicks sind graphische Elemente welche ähnliche Eigenschaften wie Windows haben.

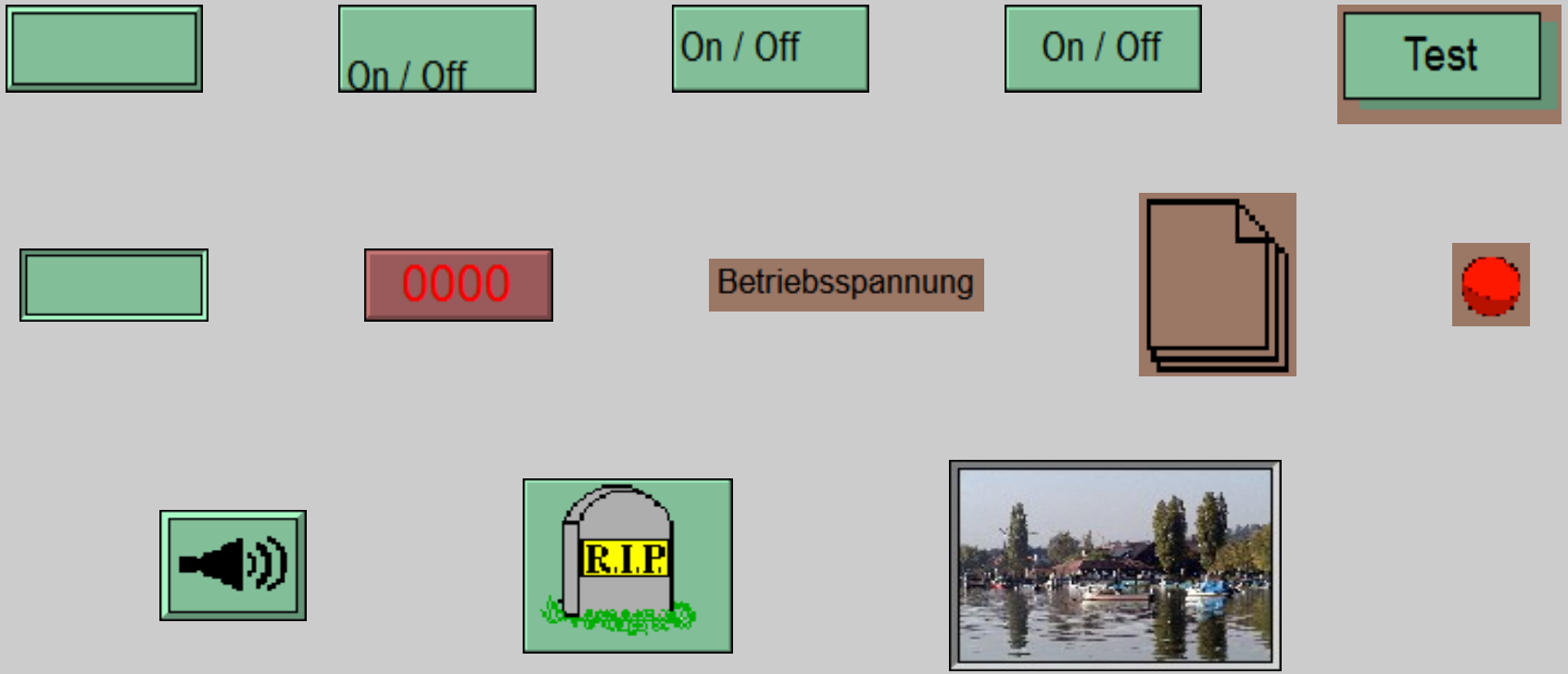
Keypicks weisen jedoch weitere Merkmale auf.

Eine wesentliche Eigenschaft ist, das ein Keypick auf einen Mausclick ein Kommando auslösen oder seine Darstellung ändern kann.

Dazu hat jedes Keypick eine normale und eine alternative Darstellung.

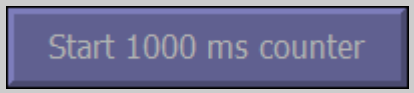
Elemente in RptOne

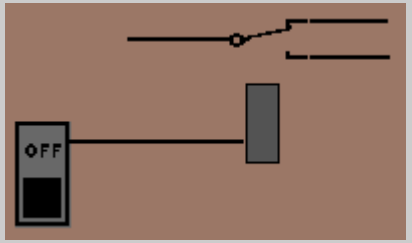
- Beispiele für Keypicks 'Mode'

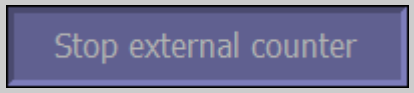


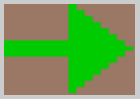
Elemente in RptOne

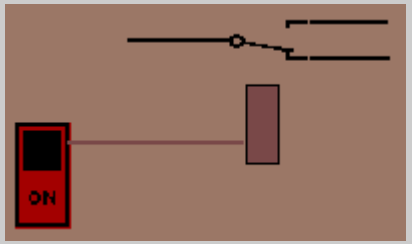
- Beispiele für alternative Darstellung eines Keypicks









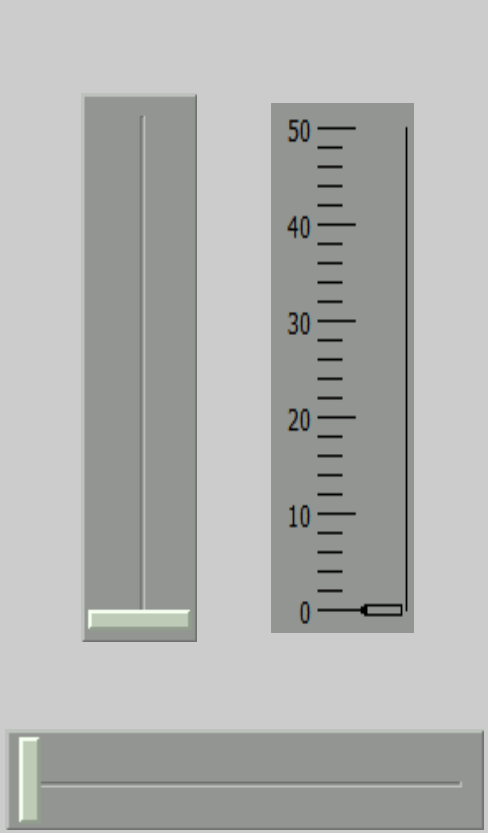


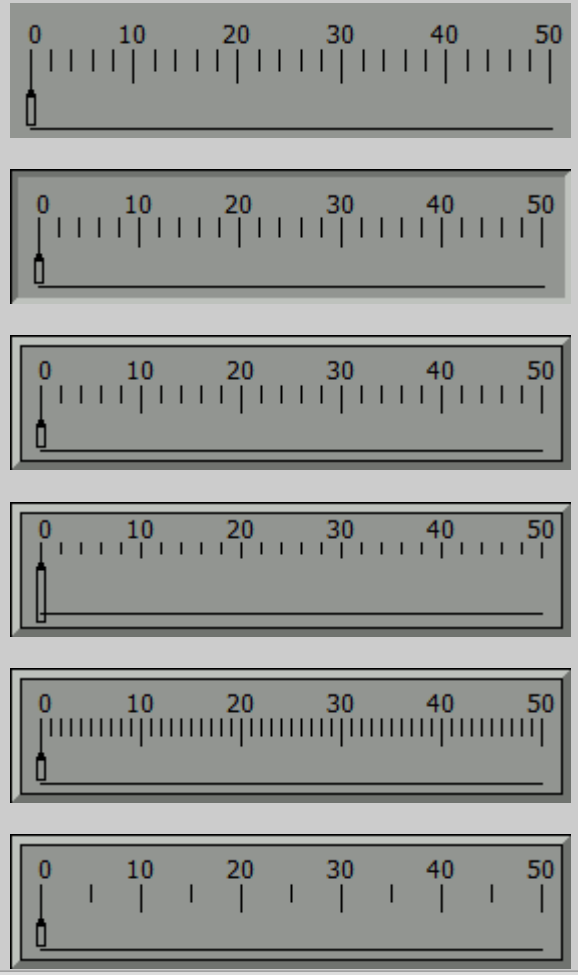
Elemente in RptOne

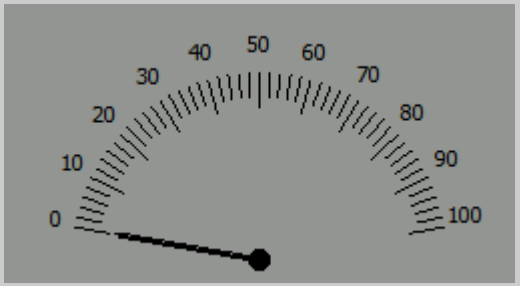
- **Objekte** stellen komplexe Elemente dar.
  - Slider
  - Instrumente
  - Diagrammfelder.

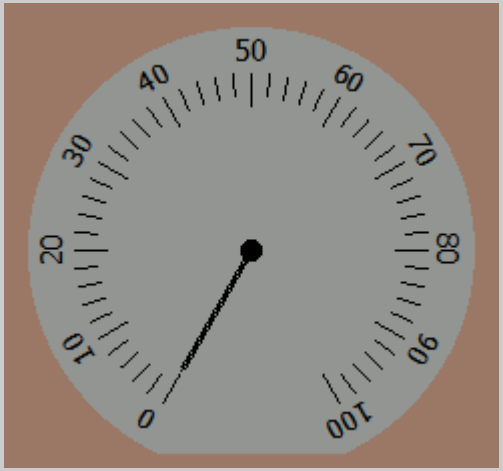
Elemente in RptOne

- Beispiele Slider und Instrumente



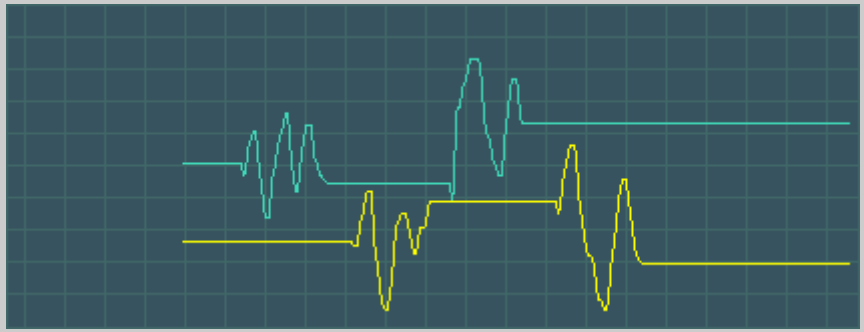


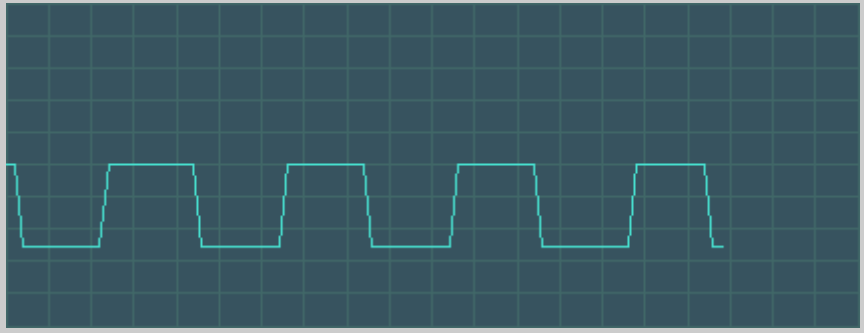


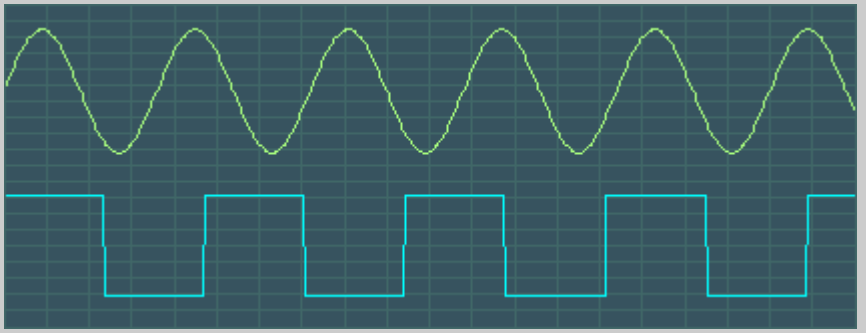


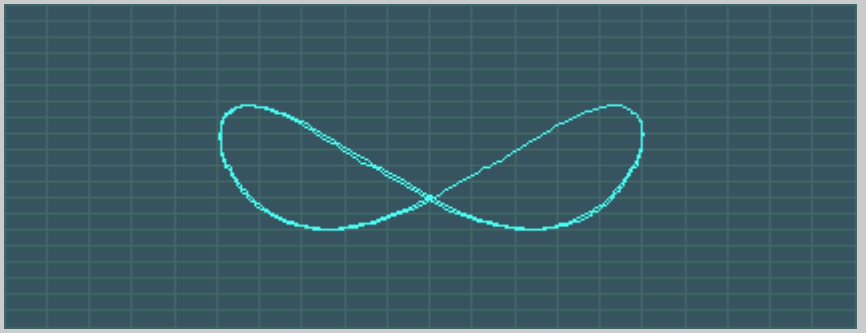
Elemente in RptOne

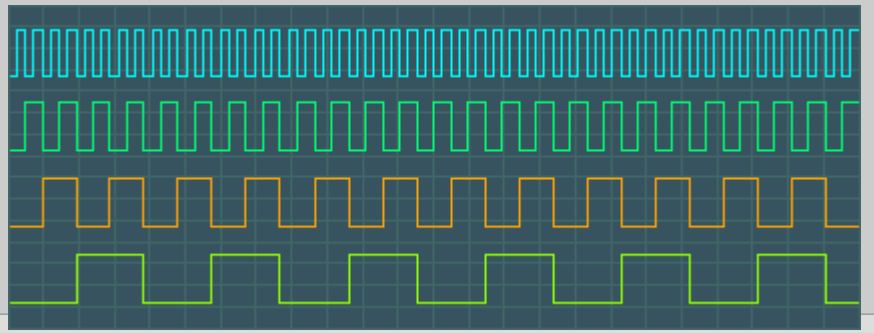
- Beispiele für Diagramme











Elemente in RptOne

- **Kommandos** sind Funktionen welche ausgelöst werden können durch:
  - Maus key down
  - Maus key up
  - Mouse key pressed
  - Applikation start
  - Application end
  - Application loop.

Elemente in RptOne

- Kommandos sind im 3 Kategorien eingeteilt:
  - System Kommandos
  - Interne Kommandos
  - Externe Programme

Elemente in RptOne

- System Kommandos
  - 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()

Elemente in RptOne

- Interne Kommandos

- `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)`

Elemente in RptOne

- Interne Kommandos

- `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)`

Elemente in RptOne

- Externe Programme
- Werden wie Konsolen Programme mit Parameterübergabe aufgerufen.
- Zur Anbindung an das RPT-Interface steht eine Library zur Verfügung.
- Ein externes Programm kann einmalig sein.
- Oder als parallele Task in einer Zeitschleife laufen.

Elemente in RptOne

- **Actions** sind Reaktionen von Keypicks auf Zustände des Programminterfaces.
- Eine Action erzwingt die alternative Darstellung eines Keypicks.

Elemente in RptOne

Die Auslösung kann durch folgende Ereignisse im Interface erfolgen:

- Bit gesetzt
- Wert kleines als Vorgabe
- Wert gleich der Vorgabe
- Wert größer als Vorgabe

Elemente in RptOne

- Des weiteren können folgende Eigenschaften eines Keypicks durch Actions verändert werden:
    - Farbe des Keypick
    - X Position des Keypick
    - Y Position des Keypick
    - Breite des Keypick
    - Höhe des Keypick.
    - Drehung des Keypick

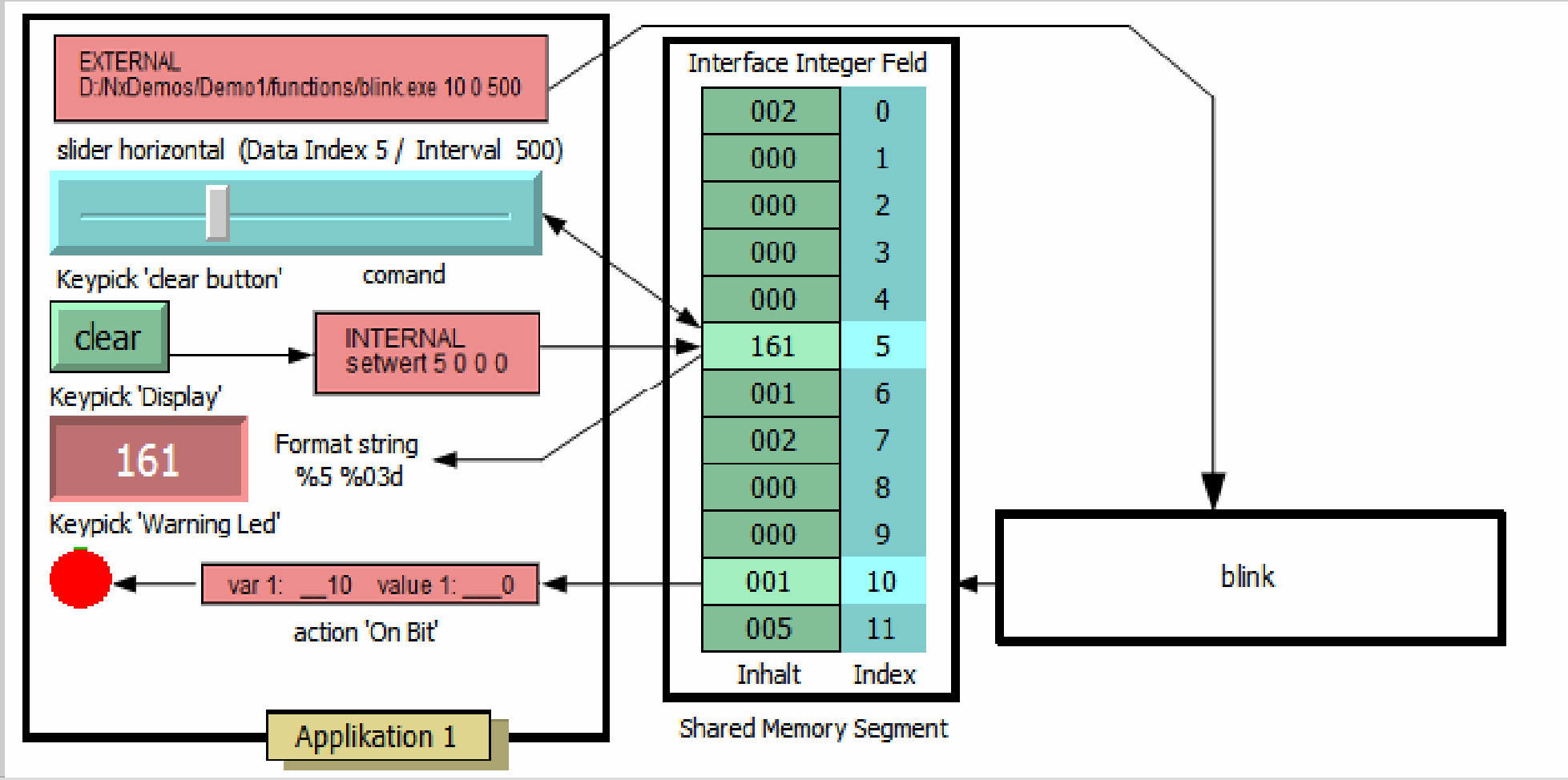
Elemente in RptOne

- RPT Interface

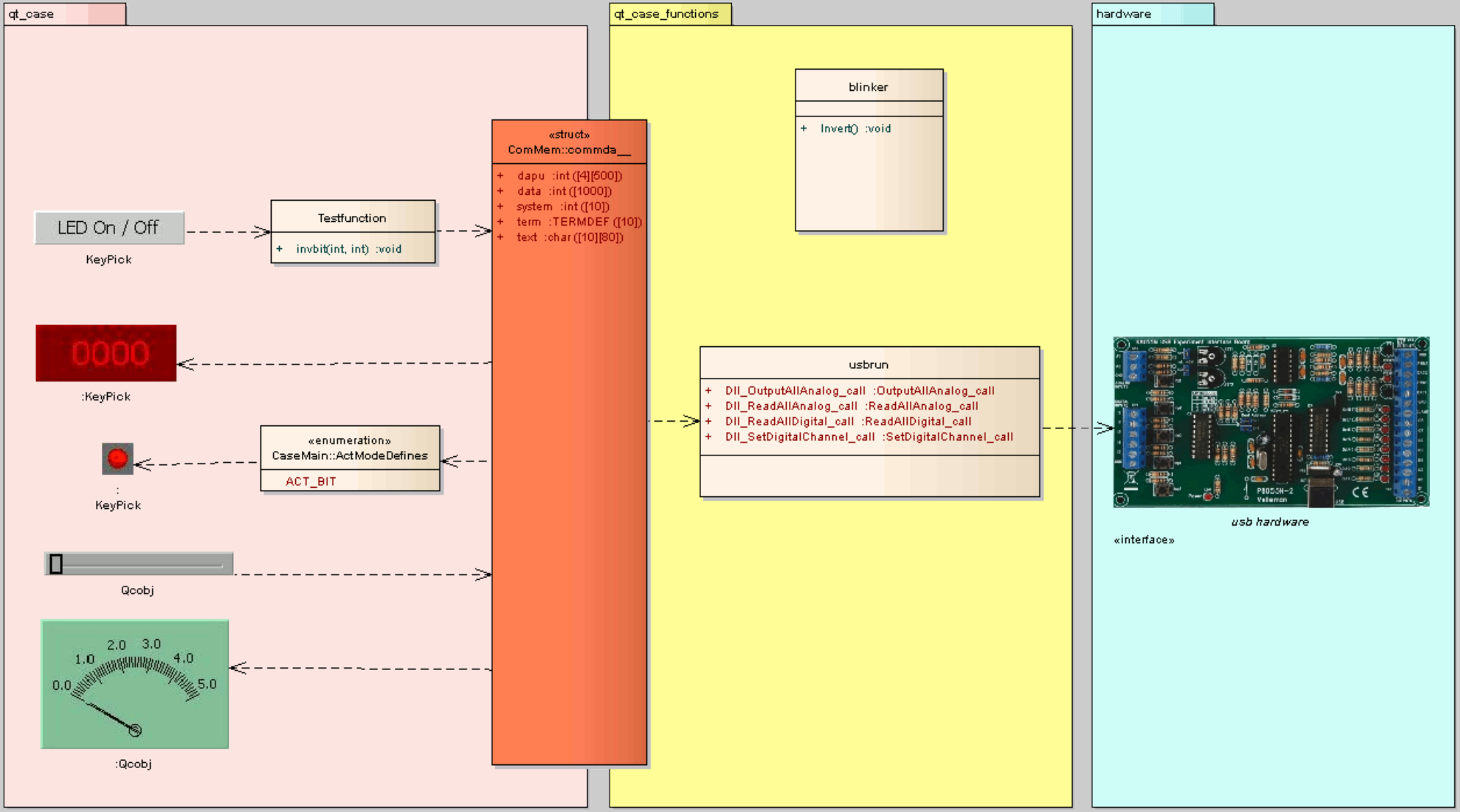
- Das Interface des Programms wird durch ein shared memory segment zur Interprocess Kommunikation gebildet.
- Auf das Interface kann intern von den Kommandos und den Aktivitäten zugegriffen werden.
- Über das Interface erfolgt der Datenaustausch zwischen den Elementen einer Applikation als auch mit den Elementen mehrerer Applikationen sowie auch der Datenaustausch mit User Programmen.

Elemente in RptOne

- RPT Interface

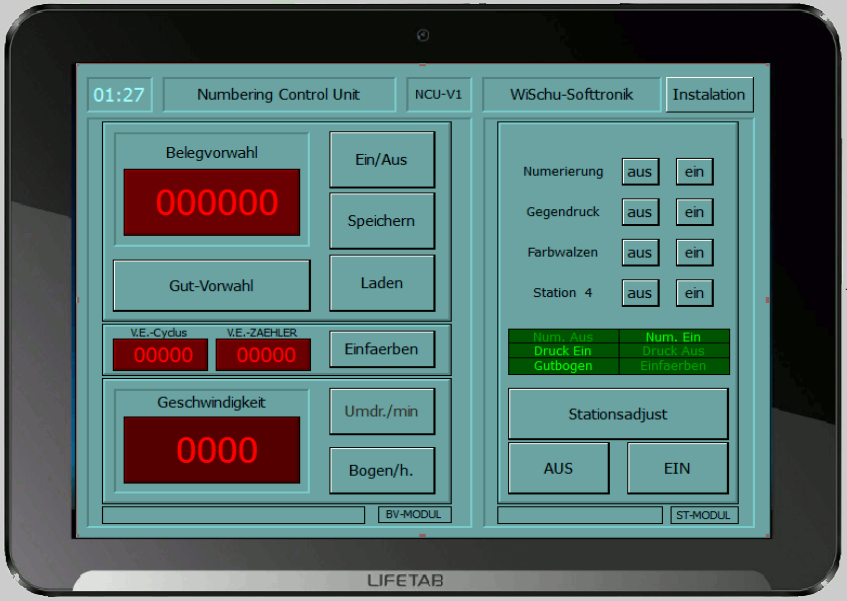


Software Struktur



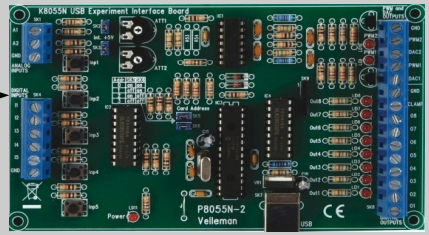
Hardware Kombinationen

Tablet Computer als
Bediener interface



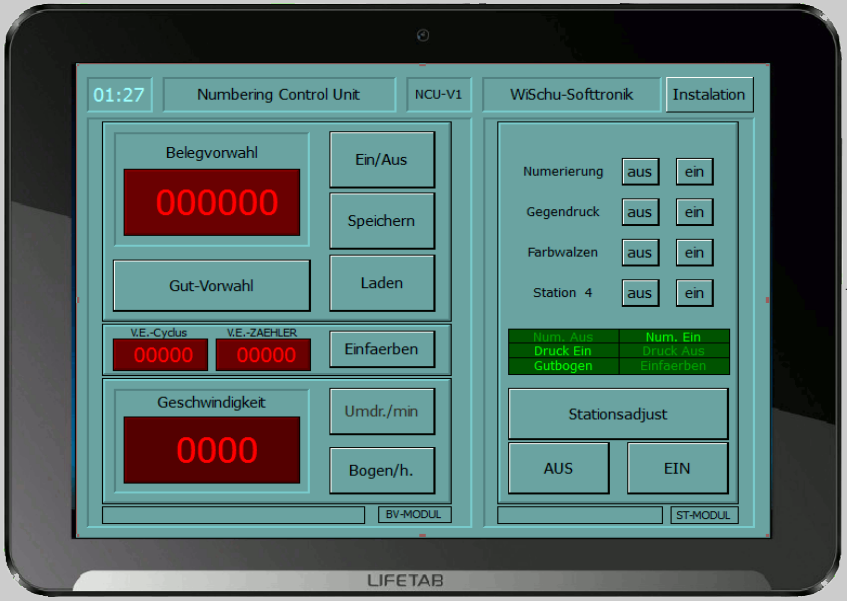
Usb-Bord als Schnittstelle zur Hardware

USB interface

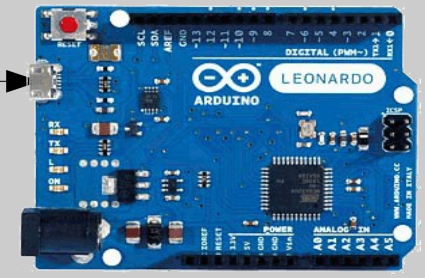


Hardware Kombinationen

Tablet Computer als
Bediener interface

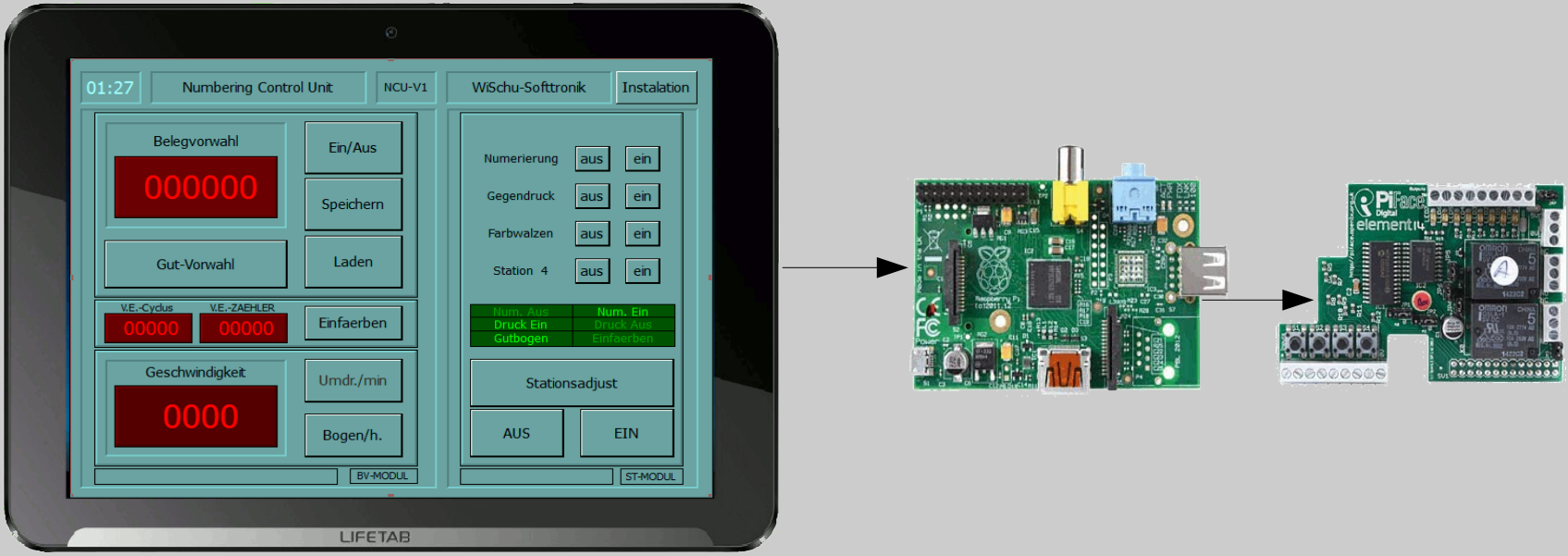


Controler mit
USB interface



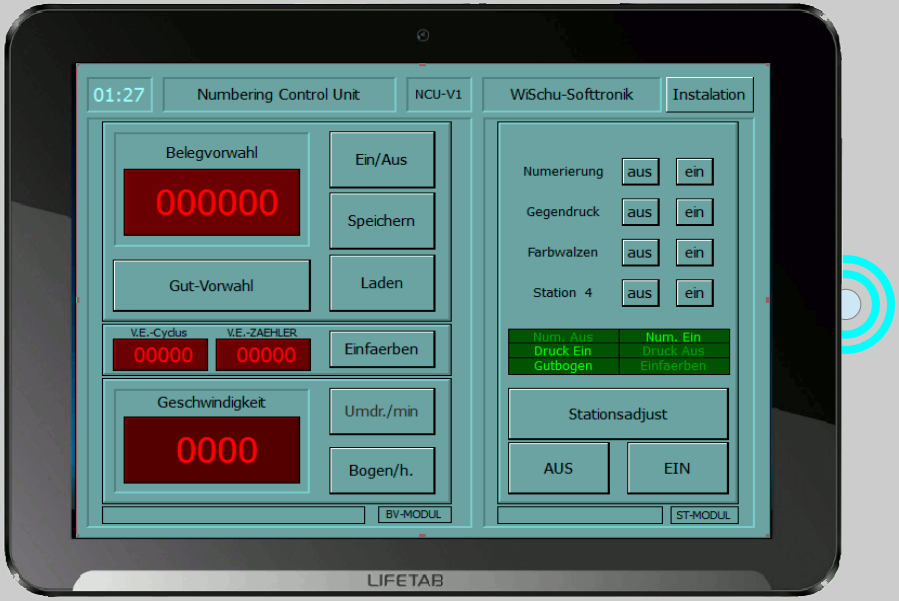
Hardware Kombinationen

Raspberry Controller mit LCD Display und Touchscreen

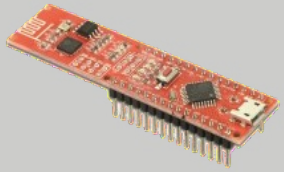


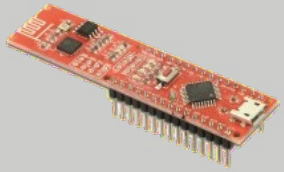
Hardware Kombinationen

Tablett Computer als Bedienerinterface



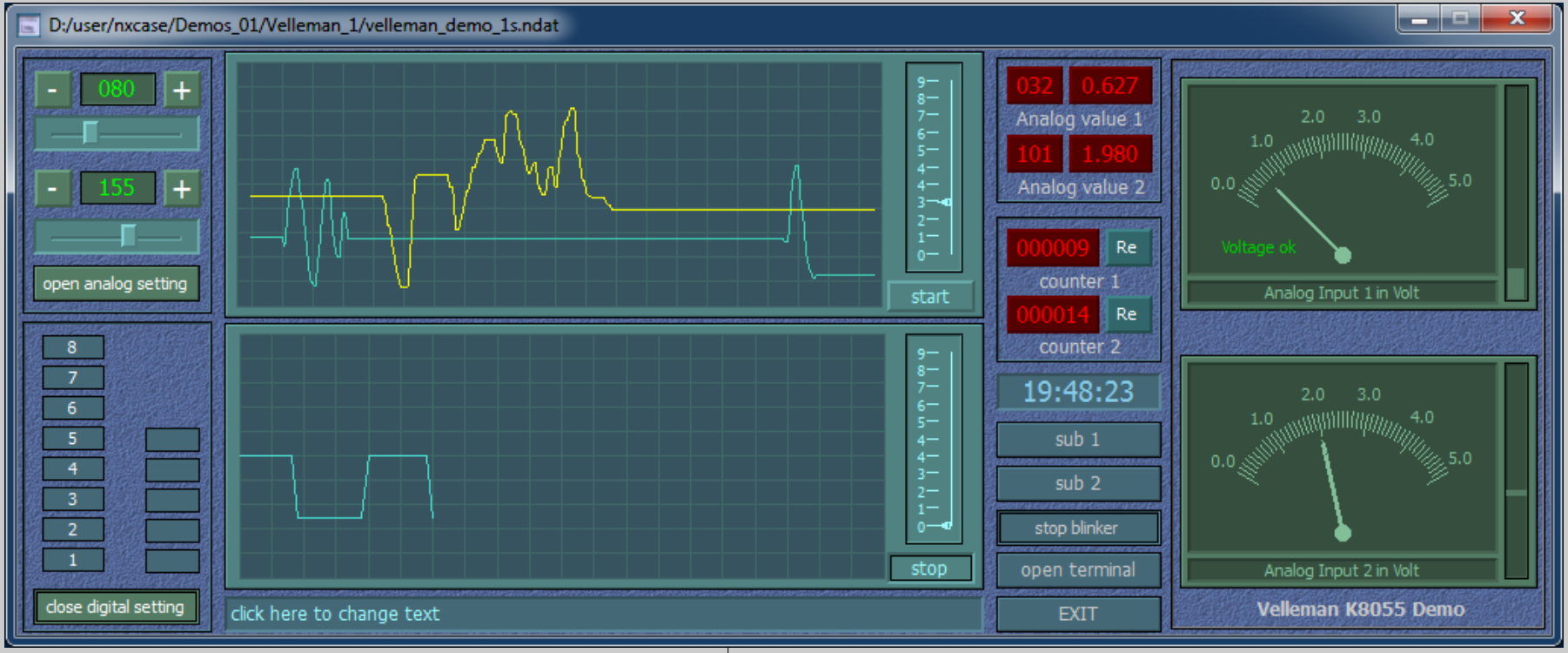
Pretzel Board - IoT WiFi Board

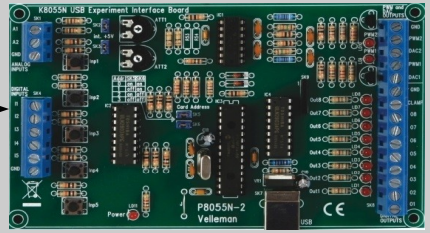




Beispiele

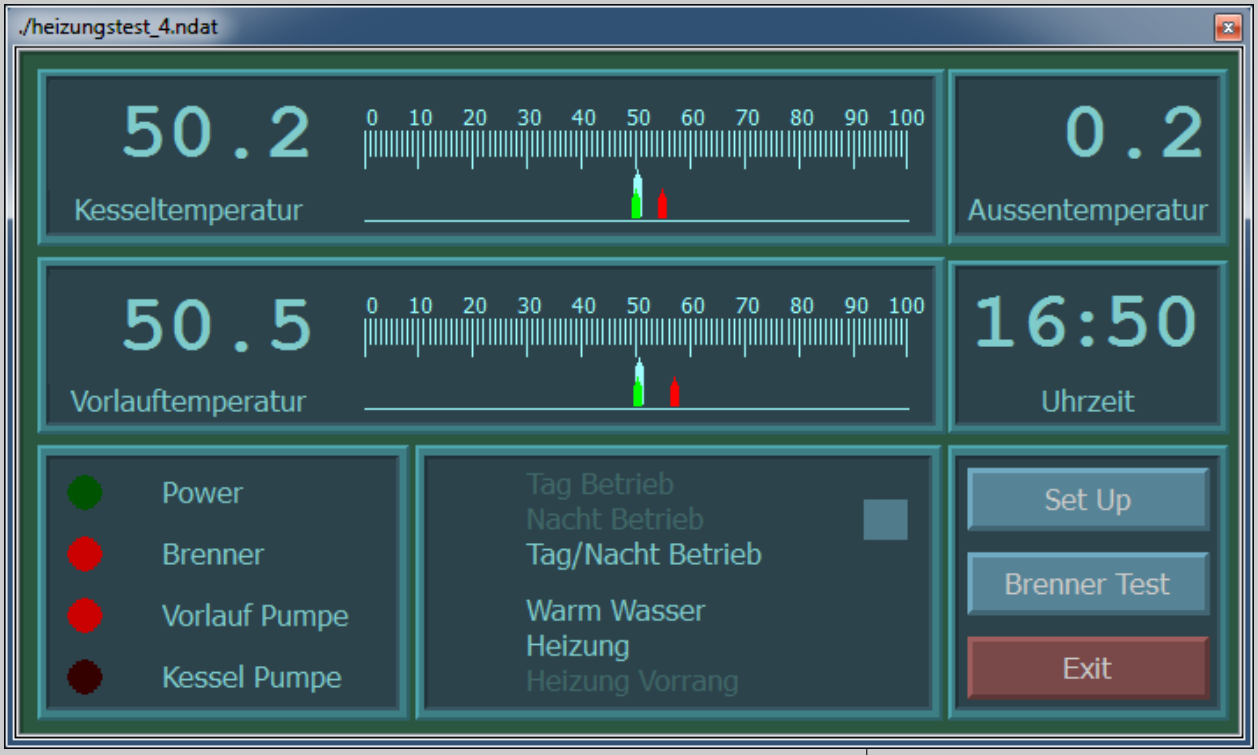
- Demo für Velleman USB IO-Board K8055



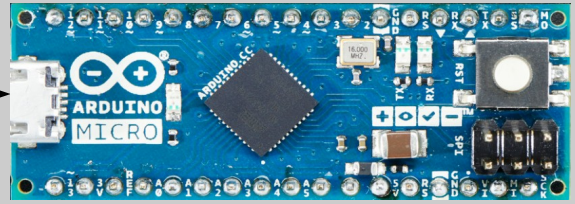


Beispiele

- Heizungssteuerung mit Arduio Micro



USB Verbindung



Beispiel User Programm 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)
{
}
```

Beispiel User Programm blink

```
#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) { }
```