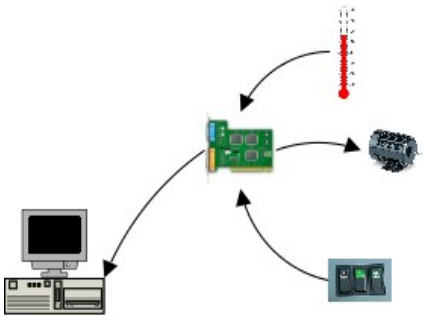
Open LabTools

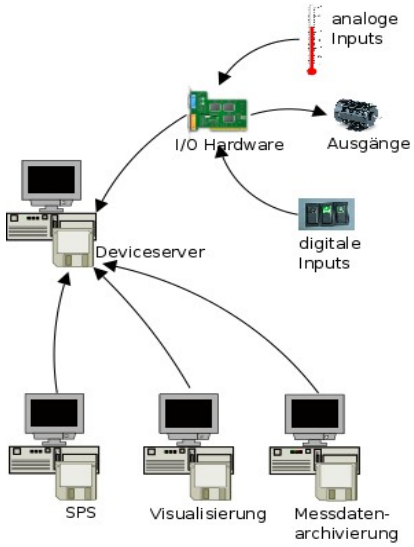
Messen, Steuern und Regeln mit dem Computer (für absolute Beginner)

Roboter, Science Fiction, moderne Industriesteuerungen und Fertigungsautomaten all das ist nur denkbar durch die Verbindung eines Computers mit Sensoren und Aktuoren, seien es Thermometer oder Motoren. Um die Werte dieser Sensoren und Aktuoren dem Computer verständlich zu machen benötigen wir eine Erweiterung der Computerhardware in Form einer I/O Karte. I/O steht dabei für Input und Output was soviel wie Eingänge und Ausgänge bedeuted. Sobald der Computer in der Lage ist verschiedenste Eingangssignale zu lesen und Ausgangssignale zu erzeugen, kann er z. B. die Werte eines Thermometers einlesen oder er kann über entsprechende Verstärker (genau gesagt sind das keine Verstärker sondern Relais [elektrisch gesteuerte Schalter]) Motoren wie Pumpen schalten. Da es in der realen Welt aber viele verschiedene Signalarten gibt, so benötigen sie auch verschiedenste I/O Karten. Zuerst einmal rein digitale I/O. Das heisst Ihre Sensoren sind Schalter die ein und aus signalisieren., bzw Ihre Aktuoren sind Relais (elektronische Schalter) mit denen Sie Lampen, Motoren etc ein und ausschalten. Daneben gibt es natürlich jede Menge analoger Werte die uns interessieren wie Temperatur, Helligkeit, Abstand oder Position. Auch dafür gibt es spezielle I/O Karten. Analoge Karten gibt es für verschiedene Messbereiche (z.B. -2,5 V bis +2,5V oder 0 bis 5V) und in verschiedenen Auflösungen (das heisst wie viele verschiedene Messwerte kann die Karte messen. Billige Wandler haben eine Auflösung von 8 Bit was 256 verschiedenen Messwerten entspricht.) Daneben gibt es auch intelligente Sensoren die, über USB oder seriellen Port angeschlossen, spezielle Werte wie Luftdruck oder Temperatur direkt liefern.



Das Bild zeigt das Prinzip: die I/O Hardware liesst analoge Werte wie die Themperatur, steuert einen Elektromotor und erkennt ob Schalter ein oder ausgeschaltet sind. Nun kann eine Software den Motor in Abhängigkeit von der Temperatur steuern/schalten. Über die Schalter werden verschiedene Temperaturen gewählt.
Für jede Mess-, Steuer- bzw. Regelaufgabe muß nun ein eigenes Programm geschrieben werden. Dabei stösst man jedoch sehr schnell auf ein Problem: die I/O Hardware kann immer nur von einem Programm zur Zeit benutzt werden. Um dieses Problem zu lösen habe ich den Device Server programmiert. Er liest und schreibt die Hardware und stellt über seine API diese verschiedensten Programmen zur Verfügung. Damit man nicht für jede Standardanwendung ein extra Programm schreiben muss, habe ich bereits einige Tools geschrieben, die alle mit dem Deviceserver zusammenarbeiten und damit ideal geeignet sind um schnelle Funktionstests zu machen. Die API des Deviceservers ist Netzwerk transparent, was bedeutet, daß die Hardware und die Software nicht zwangsläufig auf dem selben Rechner laufen müssen, sondern genauso auf verschiedenen Rechner im Netzwerk laufen können.

Da natürlich nicht alle Anwendungsfälle durch die vorhandenen Tools abgedeckt werden. Haben Sie die Möglichkeit die Device Server API auch aus eigenen Programmen zu benutzen. Dadurch sind Sie in der Lage Spezialfälle mit Ihren eigenen Programmen zu lösen und trotzdem z.B. für die Visualisierung die Webschnittstelle zu nehmen. Wie bereits erwähnt können Ihre Programme auf verschiedenen Rechnern laufen, genauso gut ist es aber möglich, dass alle Programme auf einem Rechner ausgeführt werden, der gleichzeitig auch den Deviceserver und die Hardware enthält. Diese offenen Architektur ermöglich auch die Verwendung verschiedener Betriebssysteme. So kann die Hardware und der Deviceserver unter Linux laufen, die SPS unter Windows, während die Visualisierung unter Mac OS-X ausgeführt wird. Welche Tools unter den verschiedensten Betriebssystemen zur Verfügung stehen entnehmen Sie bitte der Beschreibung der einzelnen Programme. Während alle Tools unter Linux laufen, sind bisher nur ein Teil der Tools auf andere Betriebssysteme portiert.



Zum Ende dieser kurzen Einführung sollen nun noch die Begriffe Messen, Steuern und Regeln erklärt werden.

Messen: bezeichnet die Erfassung einer realen Größe und Einordnung in eine Masseinheit, damit die Messwerte vergleichbar werden. Als Beispiel soll wieder die Temperatur dienen. Sie wird mit einem Thermometer gemessen und in der Einheit Grad Celsius angegeben. Als Statistikfreak wollen Sie die Aussentemperatur an Ihrem Wohnort das ganze Jahr über ermitteln. Wenn Sie über mehrer Jahre solche Messwerte speichern, können Sie Vergleiche anstellen die darüber Aufschluss geben welches Jahr war wärmer, verändert sich die Temperatur über die Jahre (Klimaerwärmung).

Steuern: ein Beispiel für eine typische Steuerung ist die allen bekannte Fussgängerampel. Im Ruhezustand haben die Autofahrer grün, und die Fussgänger rot. Drückt man auf den Ampelknopf, schaltet nach einiger Zeit die Ampel für Autos auf gelb, dann auf rot, danach wechselt Fussgänger auf grün, nach einiger Zeit wieder auf rot, und danach für die Autos wieder auf grün. Mit jedem betätigen des Schalters an der Ampel wiederholt sich dieser Vorgang in der exakt gleichen Art und Weise. Durch ein Signal wird eine fest vorgegebene Reihenfolge von Aktionen ausgelöst, die unabhängig von der Umwelt (Regen, Schnee oder Sonnenschein, warm oder kalt) immer in gleicher Weise erfolgt.

Regeln: die Relegelung ist im Prinzip eine Steuerung, bei der jedoch ein Signalrückkopplung stattfindet. Ein Beispiel das wohl jeder kennt, ist der Kühlschrank. Am Temperaturwähler geben sie eine Temperatur vor, die im Kühlschrank herrschen soll. Ist die gemessene Temperatur zu hoch, schaltet die Regelung die Kühlung ein, bis die vorgegebene Temperatur erreicht ist, dann wird die Kühlung ausgeschaltet bis ein zweiter Temperaturwert wieder erreicht ist bei der die Kühlung wieder eingeschaltet wird. Der ganze Vorgang ändert sich in Abhängigkeit verschiedener Faktoren. Ist die Raumtemperatur sehr niedrig wird weniger gekühlt wie bei sehr hoher Raumtemperatur. Die Umweltfaktoren werden als als Entscheidungsgrundlage in den Prozess eingekoppelt. Es ist also ein Kombination von Messen und Steuern.

Weitere Definitionen und Erklärung der Begriffe finden sie unter www.wikipedia.de unter den jeweiligen Stichworten.

Auswahl der geeigneten I/O Hardware:

Welche I/O Hardware für Sie in Frage kommt hängt von Ihren Anforderungen und Möglichkeiten ab. Wichtige Faktoren bei der Auswahl sind die Geschwindigkeit der Hardware, welche Art von Signalen (analog oder digital oder beides) müssen verarbeitet werden und natürlich das verwendete Betriebssystem. Sehr einfach in der Handhabung sind die via USB angeschlossenen Devices von BMCM. Durch den USB Anschluss müssen der vorhandene PC nicht geöffnet werden. Der Nachteil daran ist jedoch, dass die Datenübertragung über den USB Bus langsamer ist, wie bei PCI Steckkarten, die in den PC eingebaut werden.

Alle I/O Karten gemein ist der Umstand, dass die Ein- und Ausgänge nicht beliebige Signale verarbeiten können.
Ein Motor (auch ein kleiner Bastelmotor) benötigt viel zuviel Strom um direkt an einen Ausgang der I/O Karte direkt angeschlossen werden. Sie benötigen auf jeden Fall eine Relaisstufe zur Verstärkung und Entkopplung der Signale. Günstige Relaiskarten für kleine Lasten gibt es zum Beispiel bei Pollin (www.pollin.de)

Eine ganz wichtige Warnung in diesem Zusammenhang:

Netzspannung kann tödlich sein. Als Laie sollten Sie auf gar keinen Fall mit Netzspannung arbeiten. Holen Sie sich auf jeden Fall einen Fachmann zur Hilfe. Eine einfache Möglichkeit zum schalten von Netzspannungen finden Sie in der Hardwarebeschreibung des Funkmoduls, bei dem funkgesteuerte Steckdosen verwendet werden. Das bedeutet das Sie und Ihr Computer nie mit der lebensgefährlichen Netzspannung in Kontakt kommen.

Wer noch nie mit Elektronik gearbeitet hat, sollte sich zuerst mit den Grundlagen beschäftigen, damit ein Verständnis für das entwickelt wird was man macht. Wer noch nie mit Elektronik gearbeitet hat und keine Ahnung von Strom, Spannung und dem Ohmschen Gesetz hat, sollte unbedingt einen erfahrenen Fachmann zu Rate ziehen, denn kleinste Fehler reichen um die I/O Hardware oder sogar den PC dauerhaft zu beschädigen.

Eine Einführung finden Sie z.B. unter:

http://www.hcilab.org/documents/elektronik-einfuehrung.pdf	Einfacher Einführungstext
http://www.franksteinberg.de/erel.htm	Beschreibung einfacher Relaiskarte
http://www.epanorama.net/circuits/parallel_output.html	Schaltungsbeispiele parallel Port

User Guide

Draft

Im Mittelpunkt der LabTools ist der DeviceServer. Er stellt die physikalisch vorhandene Hardware den LabTools zur Verfügung. Zum Zugriff auf die Hardware benutzt der DeviceServer eine Bibliothek, die PhysMach Unit. Diese Library stellt alle nötigen Funktionen zur Verfügung um auf verschiedenartigste Hardware einheitlich zuzugreifen. Diese Lib benutzen alle LabTools Clients und sind damit je nach Konfiguration in der Lage entweder direkt oder via DeviceServer auf die Hardware zuzugreifen. Damit ist es auch möglich eine Mischung aus direkt angeschlossener Hardware und via DeviceServer zugänglicher Hardware zu benutzen. Die Möglichkeiten des DeviceServers bieten sich immer dann an, wenn mit mehreren Programmen aus der OpenLabTools Reihe gearbeitet werden soll. Dann kann eine SPS Steuerung via Web Schnittstelle bedient werden oder die Steuerung wird mittels Web 2.0 Schnittstelle (AJAX) visualisiert. Eine SPS kann mittels Objekt Erkennung auf die Realität reagieren.

Hardware

Folgende Hardware wird zur Zeit von der PhysMach unterstützt:

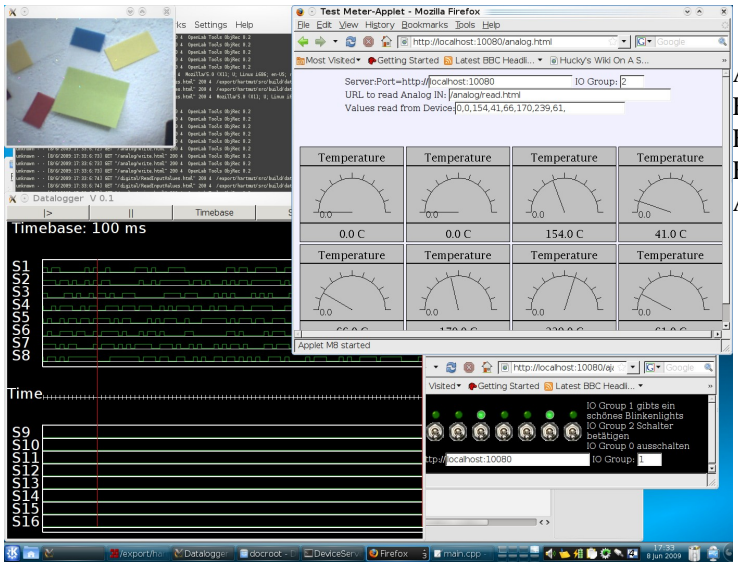
BMCM	USB-PIO, USB-AD
Code Mercenaries	IO Warrior 40
SSV	Dil NetPC
	8255 PIO ISA Bus
	Parallele Schnittstelle
	Joystick Port
Kolter Electronic	diverse I/O Karten

Folgende Resourcen werden von der PhysMach verwaltet:

128	Eingänge
128	Ausgänge
256	Merker
64	Analog Eingänge
16	Zähler
16	Timer

Die Labtools Clients

Zur Zeit stehen folgende Clients zur Verfügung:
Oszi, Datalogger und SPS sowie die Webschnittstelle zum Zugriff auf die Hardware via Browser.
In Arbeit sind die Tools ObjectRecognition zur optischen Objekterkennung via USB Camera und FunktionSolver zur Berechnung komplexer mathematischer Zusammenhänge mit den Meßwerten.
Mit diesen Tools ist es bereits möglich eine Steuerung/Regelung mit der SPS aufzubauen und die Visualisierung der Regelung mit dem Browser zu realisieren. Daneben können Oszi und datalogger zum debugging benutzt werden.



Im nebenstehenden Screenshot sehen sie die Abfrage der digitalen und analogen Signale im Firefox Webbrowser, den datalogger und die Objekt Erkennung. Unter dem Fenster der Objekt Erkennung ist ein Konsolenfenster in dem die Accesslog Datei des Deviceservers angezeigt wird.

Die Konfiguration der PhysMach

Die Konfiguration der Clients (und damit auch des DeviceServers) wird über eine Konfigurationsdatei eingestellt. Das Format der Konfiguration ist wie folgt aufgebaut:

```
# Config file for Physical Machine Lib
# pay attention, parsing of this file is set to read exact field lengthes
# only lines starting with  [DEVICE|PORT] are read, rest is discarded


# example of possible devices and their initstring
# programm PIO to A in, B out, C in
# DEVICE!P!$307:$99

# possible devices are:
#                       I = IO Warrior 40
#                       D = DIL/NetPC DNP1486
#                       P = 8255 based ISA Slot PIO
#                       L = printer port
#                       R = Random
#                       J = Joystick
#                       H = HTTP
#                       B = BMCM devices like usb pio, usb ad etc
#                       E = Exec external program

#DEVICE!P!$307:$99            # the PIO needs the config port and the configuration value
#DEVICE!I!dummy
#DEVICE!L!dummy
#DEVICE!D!dummy
#DEVICE!R!dummy
#DEVICE!J!/dev/js0
#DEVICE!H!http://www.server.com/path/read/data?§http://www.server.com/path/write/data?§
#DEVICE!B!usb-pio:/dev/ttyACM0:$ffffffff:$ffffffff:$ffffffff:
#DEVICE!B!usb-ad:/dev/ttyACM1::::
#DEVICE!E!program:param:param:


#generic config lines
#PORT![I|O|C|A|U]!iogroup!Address!devicetype
#       I=input
#       O=output
#       C=counter
#       A=analog in
#       U=analog out
```

Die Konfiguration einer USB-PIO von BMCM sieht dann so aus:

```
#BMCM USB PIO
DEVICE!B!usb-pio:/dev/ttyACM1:$ffffffff:$ffffffff:$00000000:
PORT!I!  1!   0!B
PORT!I!  2!   1!B
PORT!O!  1!   2!B
```

Die meisten Fehler sind auf die Konfigdatei zurückzuführen. Die Labtools sind im Moment noch nicht sehr aussagekräftig was Fehlersituationen angeht, teilweise werden Fehler einfach noch ignoriert. Das ist auf das frühe Stadium der Software zurückzuführen.
Um die Situation etwas zu verbessern ist das Programm PhysMachTest dabei. Mit seiner Hilfe kann die Konfigdatei sehr einfach überprüft werden. Die Konfiguration wird wie bei den anderen LabTools gelesen und interpretiert, und man ist in der Lage die Hardware manuell zu steuern.
Weitere Infos dazu finden Sie bei der Beschreibung von PhysMachTest.

Um die Konfiguration zu verstehen muss man wissen, wie die PhysMach ihre Resourcen verwaltet.

Jede von der PhysMach unterstützte Hardware arbeitet Port orientiert. Das heisst bei jeden Schreib- oder Lesezugriff auf die Hardware werden jeweils 8 Bit übertragen. Daher fasst die PhysMach auch die verwalteten Resourcen jeweils zu 8 er Gruppen zusammen. Um diese Umstände deutlicher zu machen, erkläre ich die oben gezeigte Konfiguration für die

BMCM USB-PIO.

Die Devicezeile gibt an, dass es sich um eine USB-PIO handelt, die an /dev/ttyACM1 angeschlossen ist, die Ports 0 und 1 werden als Eingänge und der Port 2 als Ausgang definiert.

Die Port Zeilen geben jetzt an, das die Ports 0 und 1 der USB PIO den Eingängen 1-8 und 9-16 zugeordnet werden. Dabei werden die Eingänge als Port 1 und 2 angegeben. Der Port 2 der PIO wird den Ausgängen 1-8, entsprechend Port 1 zugeordnet. Die PhysMach verwaltet für jede Resource eine eigene Portrange jeweils beginnend bei 1.

Konfigurationshinweise zur unterstützten Hardware:

IO Warrior 40

Alle Tests mit dem IO Warrior von Codemercs wurden mit einem einzelnen Device gemacht. Im Moment werden bis zu 8 IO Warriors unterstützt. Es sollten auch die anderen Varianten des IO Warriors unterstützt werden, bitte senden Sie mir Feedback, wenn Sie andere IO Warriors getestet haben. Da der IO Warrior alle 32 Bit in einem Zugriff lesen und schreiben kann, habe ich auch meinen Treiber dahingehend optimiert. Das bedeutet bei einem Lesezugriff auf Port 0 werden alle 4 Ports gelesen und bei einem Schreibzugriff auf Port 3 werden alle Ports geschrieben. Diese optimierte Zugriffvariante wird verwendet, wenn der Parameter Accessmethod der DEVICE Zeile auf optimized gesetzt wird. Bei jedem anderen String werden die Ports jeweils einzeln geschrieben/gelesen. Bitte setzen sie optimized nur, wenn Port 0 Eingang und Port 3 Ausgang ist.

Die Konfiguration:

```
DEVICE!I![optimized|anystring]
# erster IO Warrior 40
# der zweite IOwarrior Port 0 ist ioport=20
# der dritte IOwarrior Port 0 ist ioport=30
# der vierte IOwarrior Port 0 ist ioport=40
# port0 in,port 0 cnt, port1 in, port2 out, port3 out
PORT!I!  1!  10!I
PORT!I!  2!  11!I
PORT!O!  1!  12!I

PORT!O!  2!  13!I
```

Random Device

Das Random Device ist hauptsächlich für Tests gedacht. Für die konfigurierte IO Group werden Zufallswerte erzeugt. Die Adresse $00 ist ein Dummy der nicht verwendet wird.
Die Konfiguration:

```
# Random inputs ( for debugging )
DEVICE!R!dummy
PORT!I!  1! $00!R
```

Hier werden die Werte der Eingänge 1-8 zufällig erzeugt.

BMCM USB PIO

Im Initstring des Devices muss das Devicefile und die IO Richtung der Ports angegeben werden.
Für jeden Inputport muss $ffffffff und jeden Outputport $00000000 angegeben werden.
Die Ports werden mit 0-2 addressiert, das erste Device ist 0.
Bsp.: erster Port der 2. USB PIO = 10
dritter Port der 1. USB PIO = 2 oder 02

Die Konfiguration:

```
#BMCM USB PIO
DEVICE!B!usb-pio:/dev/ttyACM1:$ffffffff:$ffffffff:$00000000:
PORT!I!  1!  00!B
PORT!I!  2!  01!B
PORT!O!  1!  02!B
```

BMCM USB AD

Im Initstring des Devices muss das Devicefile und statt der IO Richtung der Ports einfach : als Dummy.
Weitere Docu erforderlich

HTTP Requestor

Im Initstring müssen die Read– und WriteURL angegeben werden. Die URL entnehmen Sie der DeviceServer API. Sie können für den Hostnamen statt der IP Adresse auch den Namen einsetzen, was allerdings die Zugriffszeit beeinflußt, da jedes mal die Namensauflösung ausgeführt wird. In diesem Fall bedeutet ein Ausfall der Namensauflösung auch ein Versagen des HTTP Requestors. Die Adressen der Portlines beginnen für den ersten Requestor mit 1. In unten gezeigten Beispiel bedeutet die 11 der erste Requestor, die IOGroup 1, d.h. Die Eingänge 1-8 des Deviceservers unter der IP Adresse 10.63.9.9

Die Konfiguration:

```
# HTTP Requestor
DEVICE!Hhttp://10.63.9.9:10080/digital/ReadInputValues.html§http://10.63.9.9:10080/digital/WriteOutputValues.html?§
PORT!I!  1!  11!H
PORT!O!  1!  11!H
```

Funk Modul

Das Funkmodul ist mein Devicetreiber für die von Thomas Wagner unter http://iow.wagner-ibw.de/rfctrl.html vorgestellte Hardware, angeschlossen an einen Parallel Port. Das habe ich hauptsächlich für meine eigene Verwendung codiert. Dabei wurde der Funksender wie folgt mit dem Parallelport verbunden:

LPT Bit	Funksender
Bit 0	Taste A
Bit 1	Taste B
Bit2	Taste C
Bit3	Taste D
Bit4	Funktion ON
Bit5	Funktion OFF

Bitte beachten Sie dazu die Schaltpläne und Anleitungen unter der oben genannten URL. Der IO Warrior im gezeigten Schaltplan wurde nach obiger Tabelle durch den Parallelport ersetzt.
Der von mir verwendete Treiber wird wie folgt konfiguriert:

```
# Funk Modul
# transmit time auf 1 sekunde setzen
#DEVICE!F!1000
#PORT!O!  1!$378!F
```

Im Beispiel wird die Transmittime auf 1 Sekunde= 1000 ms gesetzt. Das heisst jeder Befehl wird 1 Sekunde lang gesendet. Falls eine Steckdose nicht zuverlässig funktioniert, kann es helfen die Transmittime zu erhöhen. Es wird nur gesendet wenn eine Signaländerung wirklich vorliegt. Es ist also keine Dauersendebelastung.Als Port wird wie beim LPT Treiber die entsprechende IO Adresse des Printerports angegeben.

Kolter Electronic

Für die PCI Karten von Kolter Electronic muss zuerst der I/O Port ermittelt werden. Dazu kann man unter Linux das Kommando lspci verwenden.

Die PhysMach Unit

Die PhysMach Unit kapselt die unterschiedlichste Hardware und fasst sie zu einer „Physikalischen Maschine“ zusammen. Auf diese simulierte Hardware kann eine Anwendung über die Routinen der PhysMach zugreifen. Egal welche Hardware verwendet wird, kann die Anwendung immer auf die gleiche Art zugreifen. Dadurch wird es möglich eine Anwendung zu entwickeln, ohne über die spezifische Hardware der Zielmaschine zu verfügen. Da die Netzwerkschnittstelle wie eine Art von Hardware Schnittstelle behandelt wird, kann eine Anwendung direkt auf die Hardware zugreifen oder über die Netzwerkschnittstelle ohne das darauf speziell eingegangen werden muß.

Eine Anwendung, die ausschliesslich über die PhysMach mit der Hardware kommuniziert kann mit allen anderen PhysMach Tools zusammenarbeiten.

Die PhysMach Lib aus der Sicht eines Programmierers:

Der Programmierer greift auf die Hardware über Variablen zu, die er mit Verwendung der PhysMach automatisch definiert. Diese Variablen repräsentieren die oben genannten Hardware Resourcen.
Um eine Konsistenz und Synchronizität der Variablen mit der Hardware und damit den physikalischen Größen der realen Umgebung sicherzustellen, ruft der Programmierer die Routinen PhysMachReadDigital, PhysMachWriteDigital, PhysMachTimer, PhysMachCounter und PhysMachReadAnalog periodisch auf. Diese Routinen schreiben oder lesen die entsprechende reale Hardware und synchronisieren damit die simulierte und die reale Hardware. Aus dieser Synchronizität erwächst aber auch das Problem, dass die langsamste Hardware die mögliche Zugriffsgeschwindigkeit festlegt. Um diesem Problem zu begegnen wurde ein gänzlich anderer Zugriff auf die Hardware geschaffen, der Zugriff auf eine bestimmte reale Hardware. Damit geht logischer Weise die Synchronizität der Werte untereinander verloren. Dazu stehen dem Programmierer die Routinen PhysMachGetDevices und PhysMachIOByDevice zur Verfügung. Neben diesen Routinen benötigt der Programmierer lediglich 2 weitere Proceduren um die virtualisierte Hardware zu definieren bzw. zu initialisieren: PhysMachInit und PhysMachLoadCfg. Erstere initialisiert die virtualisierte Hardware, die 2. konfiguriert die Zuordnung von realer Hardware zu virtuialisierter Hardware.

Die Programmlogik sieht prinzipiell wie folgt aus:

Beispielprogramm (Pseudosprache) :

```
programm somethingusefull;
load PhysMach;


begin

PhysMachInit;
PhysMachLoadCfg;

while true do
      PhysMachReadDigital;
      PhysMachReadAnalog;
      PhysMachCounter;
      PhysMachTimer;

      with all this Variables do something useful
      until all values are processed,

      PhysMachWriteDigital;
endwhile

end.
```

Das Verwendungsprinzip geht aus nebenstehendem Beispiel gut hervor. Es wird zuerst die PhysMach initialisiert, dann die Konfiguration geladen. Danach wird in einer Schleife immer wieder die „Hardware“ gelesen, die Daten verarbeitet, und in die „Hardware“ geschrieben

Neben dem Zugriff über die PhysMach gibt es noch die Möglichkeit auf die Hardware über den DeviceServer zuzugreifen. Dazu kann mit jedem Browser die Hardware gelesen und geschrieben werden.

Die DeviceServer API

Im folgenden Screenshot sehen sie den Zugriff via Browser und die Objekterkennung die beide über die DeviceServer API arbeiten.

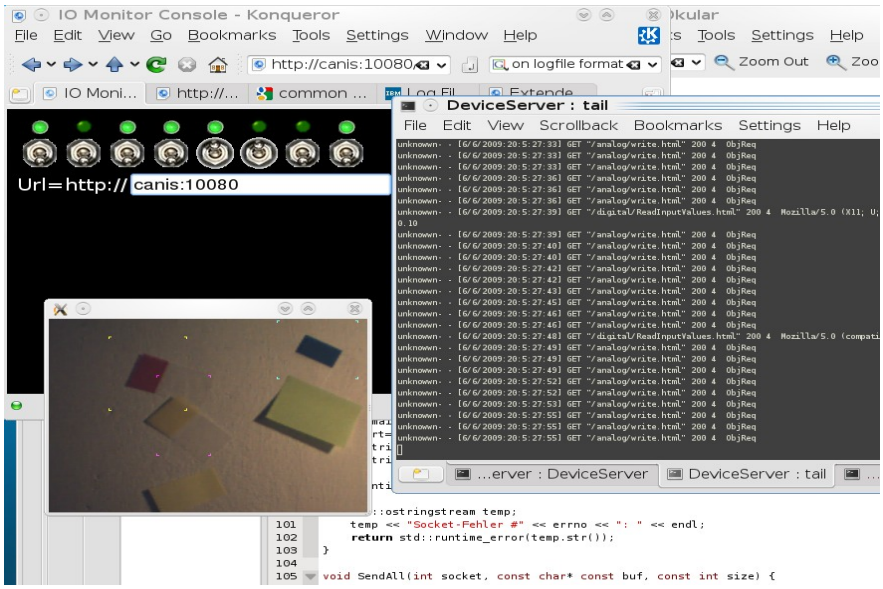


Abb1.png „Browser und Objekt Erkennung“

Die folgenden Adressen werden vom Deviceserver in der API Version 0.1 zu Verfügung gestellt:

1. /analog/read.html
2. /analog/write.html
3. /digital/ReadInputValues.html
4. /digital/ReadOutputValues.html
5. /digital/WriteOutputValues.html
6. /digital/WriteInputValues.html

Syntax und Parameter der Adressen:

/digital/ReadInputValues.html

Parameter: IO Group (1- 16)
Beispiel: http://canis:10080/digital/ReadInputValues.html?2[,3,4]
Ergebnis: <html><body> 1 0 0 1 1 1 1 0 </body></html>
Erläuterung: Es wird die IO Group 2 entsprechend den Eingängen I9 bis I16 abgefragt und das Ergebnis in Form einer minimalen HTML Seite angezeigt. Wenn mehrere IO Groups durch Komma getrennt angegeben, werden die entsprechenden IO Groups auch ausgegeben, bei 3 Ports werden dann auch 24 Bits zurückgeliefert.

/digital/ReadOutputValues.html

Parameter: IO Group (1- 16)
Beispiel: http://canis:10080/digital/ReadOutputValues.html?2[,3,4]
Ergebnis: <html><body> 1 0 0 1 1 1 1 0 </body></html>
Erläuterung: Es wird die IO Group 2 entsprechend den Ausgängen I9 bis I16 abgefragt und das Ergebnis in Form einer minimalen HTML Seite angezeigt. Wenn mehrere IO Groups durch Komma getrennt angegeben, werden die entsprechenden IO Groups auch ausgegeben, bei 3 Ports werden dann auch 24 Bits zurückgeliefert.

/digital/WriteInputValues.html

Parameter: IO Group (1 – 16), Value
Beispiel: http://canis:10080/digital/WriteInputValues.html?2,255
Ergebnis: <html><body> 1 1 1 1 1 1 1 1 </body></html>
Erläuterung: Es wird in die IOGroup 2 entsprechend den Eingängen I9 bis I16 der Wert dez. 255 entspricht binär 11111111 geschrieben.

/digital/WriteOutputValues.html

Parameter: IO Group (1 – 16), Value
Beispiel: http://canis:10080/digital/WriteOutputValues.html?2,255
Ergebnis: <html><body> 1 1 1 1 1 1 1 1 </body></html>
Erläuterung: Es wird in die IOGroup 2 entsprechend den Ausgängen I9 bis I16 der Wert dez. 255 entspricht binär 11111111 geschrieben.

/analog/read.html

Parameter: IO Group (1- 8)
Beispiel: http://canis:10080/analog/read.html?1
Ergebnis: <html><body> 120 89 0 0 0 0 0 0 </body></html>
Erläuterung: Es wird die IO Group 1 entsprechend den Analogen Eingängen A1 bis A8 abgefragt und das Ergebnis in Form einer minimalen HTML Seite angezeigt

/analog/write.html

Parameter: IDX (1 – 64),Wert1,Wert2,...,Wertn
Beispiel: http://canis:10080/analog/write.html?1,120,89
Ergebnis: <html><body> </body></html>
Erläuterung: Der Parameter IDX gibt die Nummer des ersten analogen Einganges an der geschrieben wird; die Parameter Wert1 – Wertn sind die Werte, die ab IDX in die Eingänge geschrieben werden. Bei obigem Beispiel werden die Werte 120 und 89 in die analogen Eingänge A1 und A2 geschrieben.

Je nach verwendeter Programmiersprache und Tools muß der UserAgent entsprechend gesetzt werden.
Ich habe bei der Programmierung bemerkt, daß ich bei Firefox und Mozilla ähnlichen Browsern die Antwort anders senden muß als z.B. bei Konqueror und wget. Sollten Sie also obwohl alles richtig erscheint Fehlermeldungen beim Request bekommen setzen Sie den UserAgent im Request auf Mozilla.

Bei dem http Client aus dem Gambas Package bekommen sie bei Standardeinstellungen des UserAgent den Fehler 1018 geliefert. Setzen Sie den UserAgent auf „Mozilla“ funktioniert alles wie erwartet.

Die Tools

Der DeviceServer

abstrahiert die Hardware und stellt den Tools die Hardware als Server zur Verfügung.
Dazu wird der in den DeviceServer integrierte webserver verwendet. Zum debugging steht ausserdem eine Telnet Schnittstelle zur Verfügung.

Nach der Konfiguration des DeviceServers über die Datei DeviceServer.cfg, kann der DeviceServer durch Eingabe von ./DeviceServer gestartet werden.
Die Startmeldungen des DeviceServers können je nach Konfiguration z.B. so aussehen:

```
$ ./DeviceServer
detected Hardware: ,Random
starting DeviceHandler for Device:R
started Device Handler Thread..1
Starting Webserver Thread...
started Webserver Thread, going to start Server...
Webserver started, ready to serve
Starting Telnet Thread...
started Telnet Thread..3
Starting Statistics Thread...
```

Durch Eingabe von e kann der DeviceServer beendet werden, die Stopmeldungen sehen bei obigem Beispiel so aus:

```
Device Handler going down..1
Webserver going down..
Telnet Accept error: 22
$
```

Wenn der DeviceServer gestartet ist kann man mittels Telnet sich mit dem DeviceServer verbinden und neben Statisticdaten auch denn Zustand der Hardware abfragen:

```
$ telnet localhost 45054
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Welcome to Device Server Monitor, use "close" to quit
>s
DeviceHandler R  44153787=2609515 loops/second
Webserver  1=0 loops/second
Telnet Thread  0=0 loops/second
Stats Thread  2=0 loops/second

>h
cmd lines are build like this
cmd hardware number [value]
cmd=[R|W|H|E|C|S] for read, write, help, end, config and stats
hardware=[C|I|O|A] for counter, input, output, analog values
Number = Number of line
Value= value needed when writing lines
```

Durch einen Programmfehler kann es passieren, daß sich der DeviceServer beendet, wenn man eine Telnet Session über Strg+Alt] und quit beendet. Um diesen Fehler zu umgehen sollten Sie die Telnet Session mit `close` beenden.

Der Deviceserver schreibt alle Zugriffe in eine Logdatei im Common Logfile Format, das bedeutet Sie können Standard Statistic Tools anderer Webserver benutzen um die Zugriffe auszuwerten.

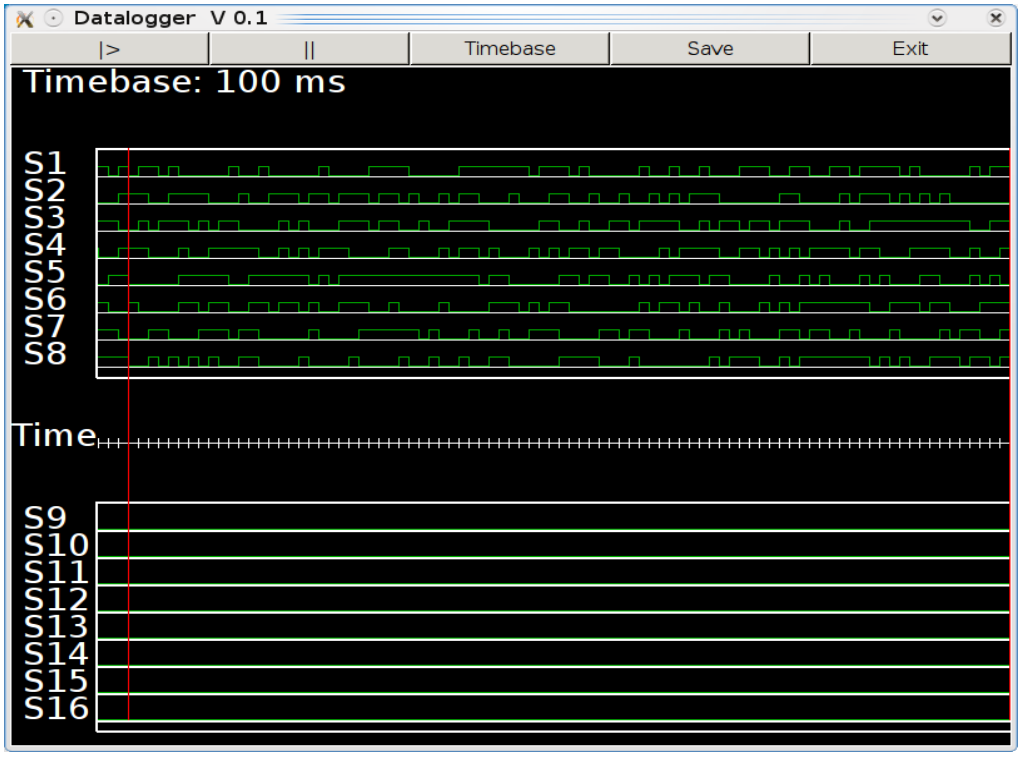
Der Deviceserver kennt keine Zugriffscontrolle und bietet keinerlei Vorkehrungen gegen unbefugten Zugriff. Daher sollte der DeviceServer nur in gesicherten Umgebungen eingesetzt werden.

In der Standardkonfiguration des DeviceServers ist lediglich ein Random Number Generator auf der IO_Group 1 der Eingänge (die Blinkenlights in den Screenshots).

Der Port des Webserver ist 10080 und der Telnet Schnittstelle 45054. Diese Ports können zur Zeit noch nicht geändert werden.

Der Datalogger

Der Datalogger kann 16 digitale Kanäle auf einer Zeitachse anzeigen. Die Konfiguration erfolgt über die Datei .datalogger.cfg. Über die Buttons kann die Messung gestartet und pausiert werden, die Timebase geändert werden und der Datalogger beendet werden.



Die obige Abbildung zeigt bei den Eingängen S1-S8 die vom DeviceServer exportierten Zufallszahlen.

SPS

SPS4Linux oder OpenSPS ist eine Softsps und ein integraler Bestandteil der OpenLabTools. Im einzelnen besteht es aus sps der IDE zur komfortablen Entwicklung und Test von Steuerungen und run_sps dem reinen Laufzeitinterpreter, der benutzt wird um fertige Steuerungen ablaufen zu lassen. Beide Tools werden über die gemeinsame Konfigurationsdatei .run_sps.cfg kontrolliert. Auch hier steht der volle Konfigurationsumfang zur Verfügung, womit sie in der Lage versetzt werden netzwerktransparente Steuerungen zu entwickeln und via Browser zu steuern und

visualisieren. Weitere Informationen finden Sie in der Docu von OpenSPS die sie in der IDE lesen können. Diese Docu finden Sie in der Datei sps.doc, einem Textdokument, dass Sie mit einem Editor öffnen können bzw. auch ausdrucken können.