

KOMMUNIKATION MIT EXTERNER HARDWARE

Carsten Knoll, Institut für Regelungs- und Steuerungstheorie, TU Dresden

Dresden, 27.01.2013

- Ziel:
 - Schnittstellen ↔ Module
 - Empfehlungen
- Aufbau:
 - Einleitung**
 - Schnittstellen**
 - Serielle Schnittstelle
 - parallele Schnittstelle
 - GPIB
 - Ethernet
 - DLL-Treiber nutzen**
 - Anwendungsbeispiele**
 - USB Missile Launcher
 - Mobiler Roboter (Arduino-Plattform)
 - Umsetzungshinweise**

Vorbemerkungen

Einleitung

Schnittstellen

Fertige Treiber Nutzen (DLLs)

Anwendungsbeispiele

Umsetzungshinweise

- Messgeräte
- Funktionsgeneratoren
- Steuerungen für Positioniertische
- Optische Komponenten (Lampen, Laser, Filter, ...)
- Temperaturregler
- Mikrocontroller („ μ C“)
- Anderer Rechner
- ...

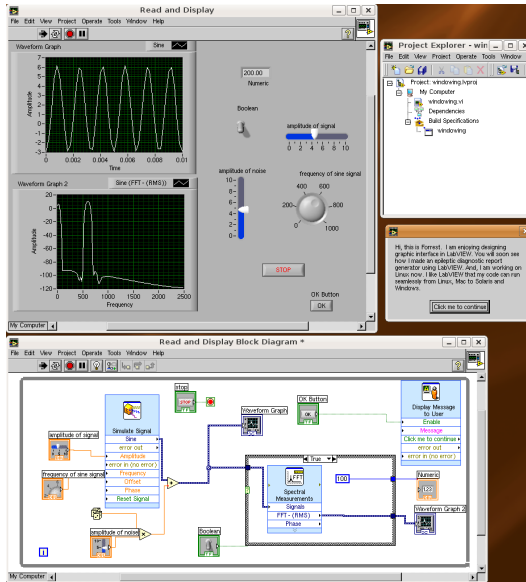
- Messgeräte
 - Funktionsgeneratoren
 - Steuerungen für Positioniertische
 - Optische Komponenten (Lampen, Laser, Filter, ...)
 - Temperaturregler
 - Mikrocontroller („ μ C“)
 - Anderer Rechner
 - ...
-
- Hintergrund: Eigenen Erfahrungen ( Fraunhofer_{ILT},  iapp)

- Platzhirsch: LabVIEW (National Instruments)
- Umfangreiche (Treiber-)Bibliothek
- Datenfluss orientiert → grafisches Programmieren
- Intuitive Parallelisierung, sehr einfaches Erzeugen von GUIs

- **Nachteile** (persönliche Meinung!)
- Haupteingabe per Maus (= Informations-Nadelör verglichen mit Tastatur)
- Modularisierung/Kapselung aufwendiger
- Tendiert schnell zu Unübersichtlichkeit
- Permanente Platzprobleme → Spazwang auch bei Kommentaren
- Wartbarkeit ↘, Erweiterbarkeit ↘
- ...



So soll es aussehen





Wann ist Python eine mögliche Alternative?

- wenn keine (fast) fertige LabVIEW-Lösung existiert
- wenn keine speziellen LabVIEW-Features benötigt werden (FPGA, harte Echtzeit)
- wenn aufwendige Algorithmen zu implementieren sind (nicht nur Daten verschieben)
- Lizenzkosten eine Rolle spielen

- es existieren „harte“ und „weiche“ Echtzeitbedingungen
- Hart: Verletzung nicht hinhnehmbar (Steuerung der Flugzeughydraulik)
- Weich: Verletzung unschön aber nicht tragisch (DVD-Player)
- Ca. 95% der Anwendungen: weich
- Hauptkriterium: Determiniertheit
- Python nicht-deterministisch (autom. Garbage-Collector)
→ nur weiche Echtzeit
- Ausreichend schnell? → abhängig von Aufgabe

Vorbemerkungen

Einleitung

Schnittstellen

Fertige Treiber Nutzen (DLLs)

Anwendungsbeispiele

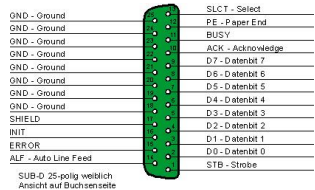
Umsetzungshinweise

- „Seriell“: Daten (Bitfolge) werden hintereinander übertragen
- Meist RS-232 gemeint
- Heute oft über USB emuliert (Universal Serial Bus)
- Windows: COM1, ... Unix: /dev/ttyS1...
- Viele (Mess-)Geräte damit ausgestattet
- Paket serial (Projekt „pySerial“)



```
import serial
ser = serial.Serial(0, 19200, timeout=2.5) # open first
      serial port
print ser.portstr           # check which port was really used
ser.write("hello")          # write a string
response = ser.readline()   # wait for response
ser.close()
```

- „Parallel“: Mehrere Bitfolgen gleichzeitig über parallele Leitungen übertragen
- Auch als „Druckerausgang“ bekannt
- Einfachste Möglichkeit für digitale E/A
- Heute kaum noch vorhanden
- Empfindlich gegen Überspannung und Kurzschluss
- Paket parallel (Projekt „pySerial“)



```
import parallel
p = parallel.Parallel()          # open LPT1
p.setData(0x57) # write 0101 0111
responseBit = p.getInPaperOut()
```

- General Purpose Interface Bus
(auch:HP-IB, IEC-625-Bus,...)
 - Bis zu 15 Geräte parallel
-
- Paket visa (Projekt „pyvisa“)



```
import visa
keithley = visa.instrument("GPIB::12")
ident = keithley.ask("*IDN?")
assert ident.startswith('KEITHLEY INSTRUMENTS INC.') #
    Konsistenztest
keithley.write(':CONF:VOLT:DC')
v = keithley.ask(':READ?')
print float(v)
```

Ethernet (Netzwerk, Sockets)

- Modernere Geräte mit Netzwerkanschluss
- Interner Webserver für die Kommunikation
- Socket (engl. für „Sockel“):
 - = Endpunkt einer Netzwerkverbindung
- → Client-Server-Architektur
- Paket socket (python standard libray)



```
# Echo client program
import socket

HOST = '141.30.61.152'    # The remote host
PORT = 50007              # The same port as used by the
                           server
s = socket.socket()
s.connect((HOST, PORT))
s.send('Hello, world')
data = s.recv(1024)
s.close()
print 'Received', repr(data)
```

Vorbemerkungen

Einleitung

Schnittstellen

Fertige Treiber Nutzen (DLLs)

Anwendungsbeispiele

Umsetzungshinweise

- Gerätetreiber vom Hersteller oft als kompilierter Code mitgeliefert (.dll („Dynamic Link Library“) oder .so; „Shared Object“)
- Paket ctypes (python standard library)
- Benötigt: Kenntnis über Namen u. Signatur der Funktionen
- Beachtung der Datentypen erforderlich

```
# Optischen Zeilen-Sensor (Linien-Kamera) auslesen
# (Kommunikation ueber USB mit unbekanntem Protokoll)

import ctypes

camdll = ctypes.windll.LoadLibrary('D:/devices/cam123.dll')
camdll.setIntegrationTime(5) # 5ms

valuearray_type = ctypes.c_ushort * 512
valuearray = valuearray_type()
pixels = camdll.ReadData( ctypes.byref(valuearray) )
assert pixels == 512 # Konsistenztest
values = list(valuearray) # ctypes array -> python list
```

Vorbemerkungen

Einleitung

Schnittstellen

Fertige Treiber Nutzen (DLLs)

Anwendungsbeispiele

Umsetzungshinweise

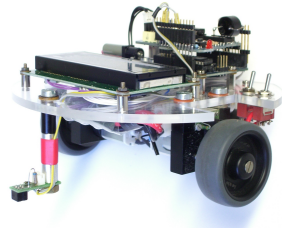
- Aufgabe: Lichtstrahl abschatten (ohne zeitliche Anforderungen)
- Professionelles Equipment: >500€
- Idee: Zweckentfremden von ... Spielzeug
- → Zweiachsig einstellbarer Raketenwerfer
- Steuerung vom PC aus mittels Windows-GUI-Programm
→ egal
- ∃ DLL ⇒ ∃ python-wrapper
- Aufgabe gelöst



- Projekt für Schülerpraktikum (7./8. Klasse)
- Kann: Fahren, Blinken, Hupen, „Sehen“ (Helligkeitssensoren), kommunizieren (Display, RS-232)

Arduino:

- Open-Source μ C-Entwicklungsplattform einfach und leistungsfähig
- Atmel AVR-Mikrocontroller + Platine
- USB, Spannungsversorg., LEDs, Reset-Knopf
- Plug'nPlay
- Digital + Analog I/O (AO = PWM),
- Software: IDE + Bibliotheken + Beispiele
- → Stark vereinfachte C++-Programmierung



Ziel:

- Vom PC aus den Roboter steuern

Umsetzung:

- C++-Programm auf μ C wartet auf Befehle und führt dann bestimmte Aktionen aus
- Python Programm um die Befehle zu senden
- Jeder Befehl besteht aus zwei Byte:
 - Kommando ('F'=Forward, 'B'=Backward, ...)
 - Argument (numerischer Wert des Bytes, [0, 255])
- Einbindung von `ipython` $\Rightarrow$ interaktives Ausprobieren

Vorbemerkungen

Einleitung

Schnittstellen

Fertige Treiber Nutzen (DLLs)

Anwendungsbeispiele

Umsetzungshinweise

- Automatisierungsaufgabe schriftlich durchdenken
- Möglichst viel bestehendes nutzen
- Objektorientierung: ein Gerät $\leftrightarrow$ eine Klasse ($\leftrightarrow$ eine Datei)
- Modularisierung: Eigenes Paket („devices“)
- Initialisierungsphase und ggf. ordnungsgemäßes Abschalten
- Simulationsmodus vorsehen (Programmtest ohne angeschlossene Geräte)
- Konsistenztest (Überprüfen eigentlich bekannter Informationen)
- Überprüfung zulässiger Wertebereiche
- Logging-Funktionalität

- <http://pyserial.sourceforge.net>
- <http://pyserial.sourceforge.net/pyparallel.html>
- <http://pyvisa.sourceforge.net/>
- <http://docs.python.org/library/socket.html>
- <http://docs.python.org/library/ctypes.html>

- <http://www.arduino.cc>
- <http://www.arduino.cc/playground/Interfacing/Python>

- <http://wiki.python.org/moin/BitwiseOperators>