Technische Universität Kaiserslautern
Institute of
Integrated Sensor Systems
Dept. of Electrical Engineering and Information Technology

"In-the-loop-learning"-Implementierung für analogen Neurochip 'Silimann 120cx'

Max Reichardt

11. September 2006

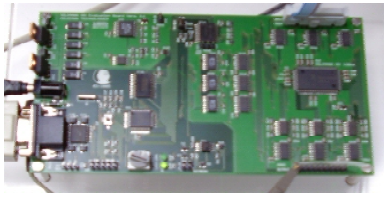
Prof. Dr.-Ing. Andreas König

Inhalt

- Aufgabenstellung
  - Probleme
- Implementierung
  - in LabView
  - mit C#
    - Board-API
    - Systemarchitektur
- Bewertung & Ausblick

Aufgabenstellung
Trainings-Software anpassen

- Ausgangslage:
  - Beim Lernen wird Modell zur Berechnung der KNN-Ausgaben verwendet
  - Schnell, aber möglicherweise ungenau
- Aufgabe:
  - Ausgaben sollen direkt mit Hilfe des Boards ermittelt werden
  - Software entsprechend anpassen



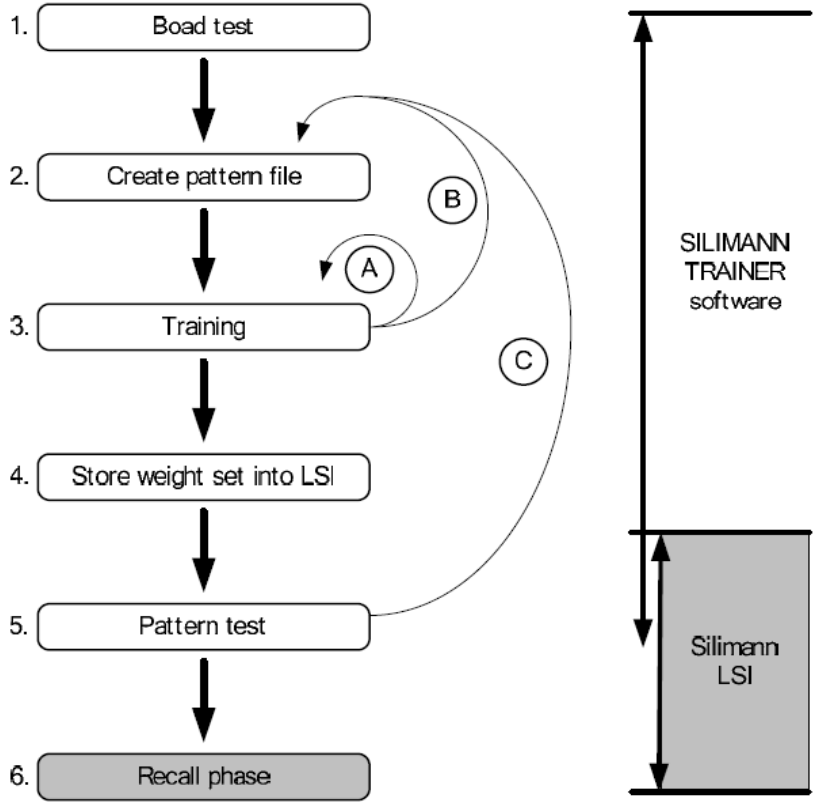
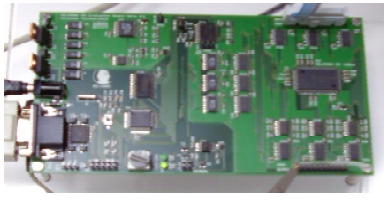


Figure 6. Neurokit workflow

Aufgabenstellung
Trainings-Software anpassen

- Ausgangslage:
  - Beim Lernen wird Modell zur Berechnung der KNN-Ausgaben verwendet
  - Schnell, aber möglicherweise ungenau
- Aufgabe:
  - Ausgaben sollen direkt mit Hilfe des Boards ermittelt werden
  - Software entsprechend anpassen



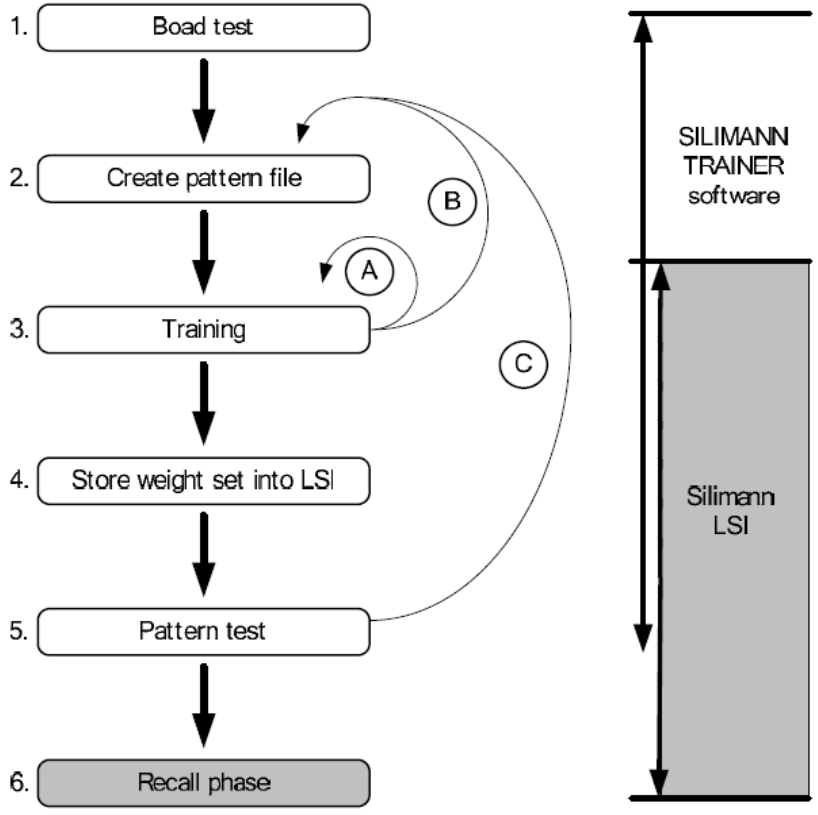
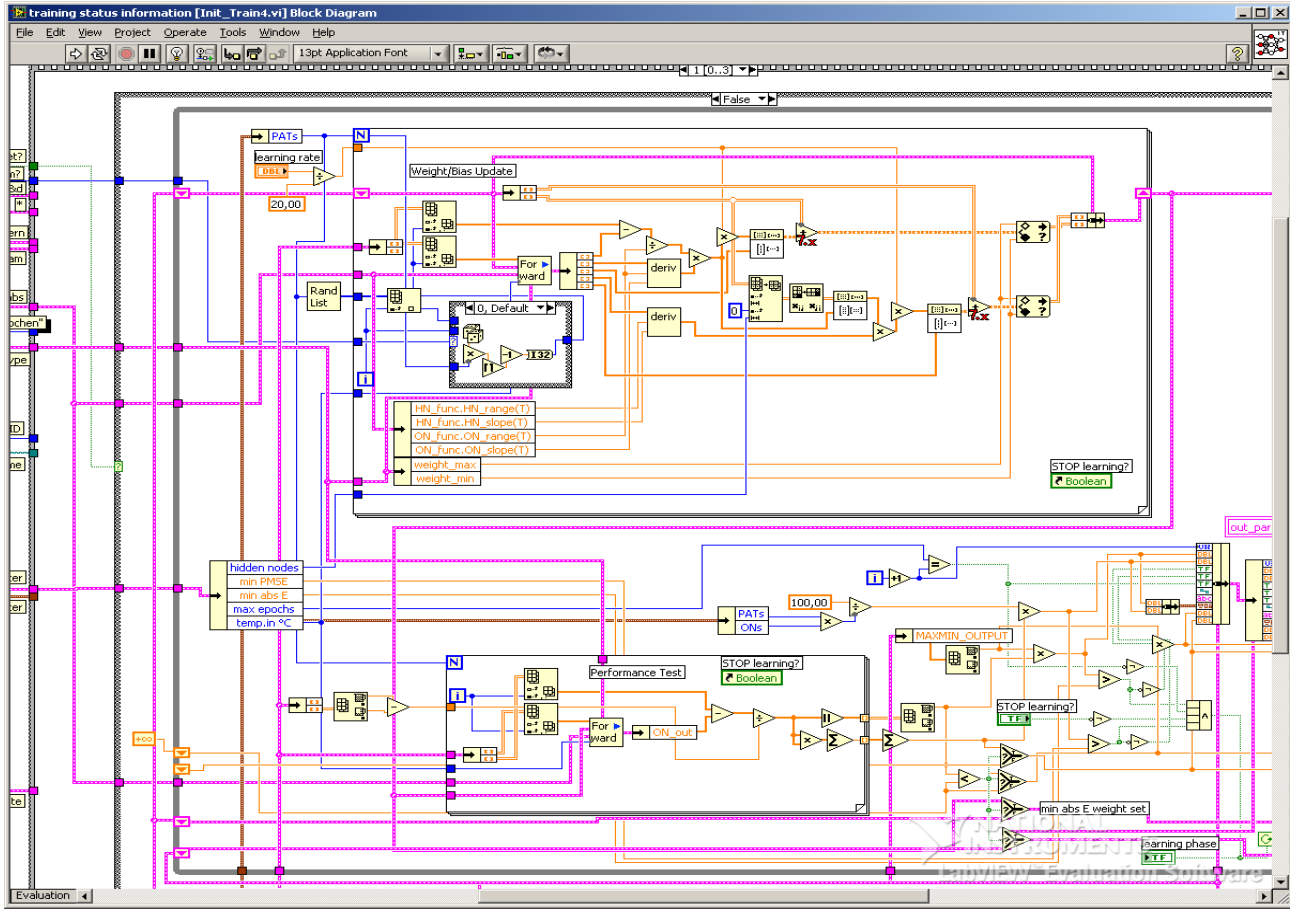
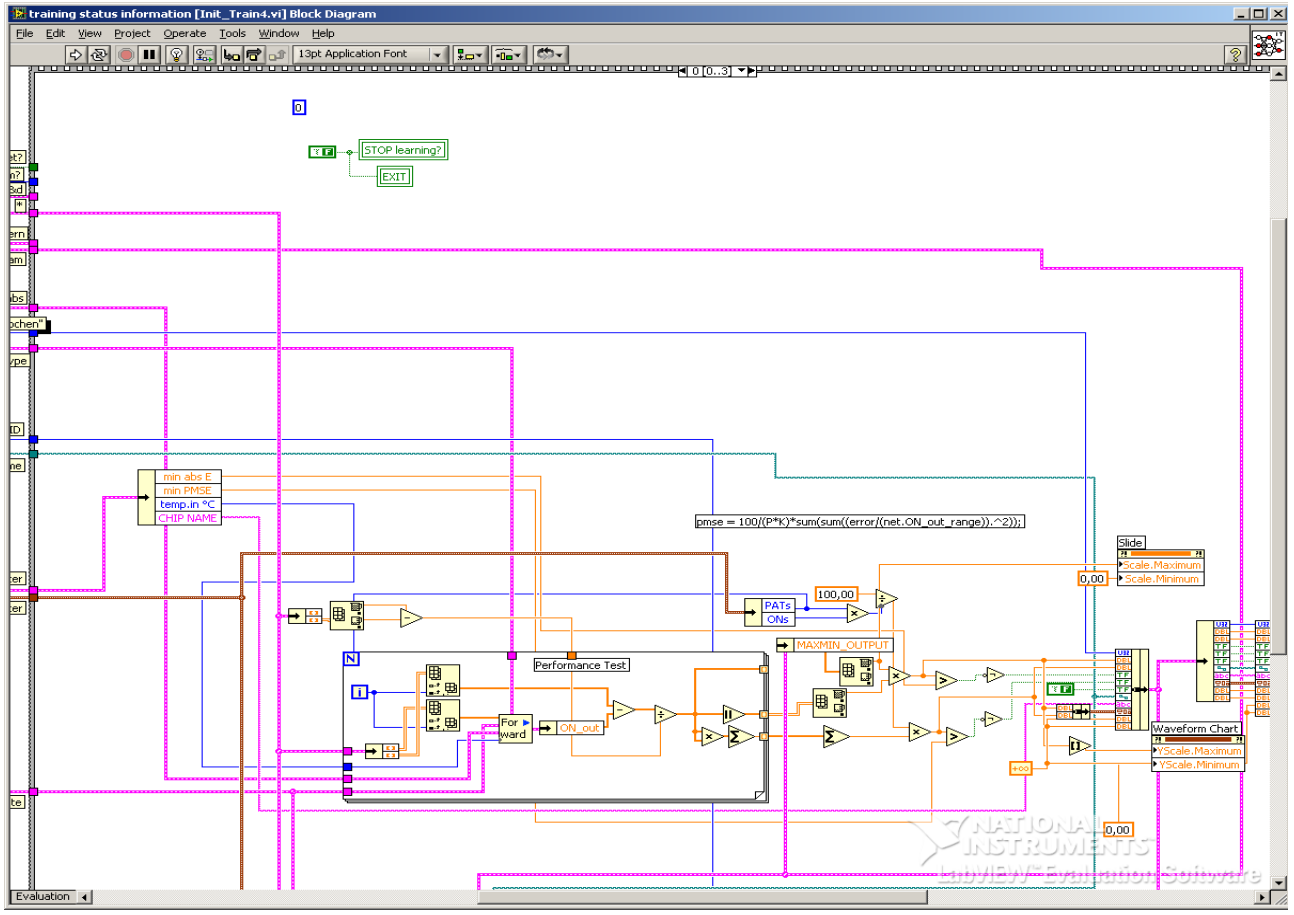


Figure 6. Neurokit workflow *

Schwierigkeiten (1)
Trainings-Software komplett in LabVIEW „geschrieben“

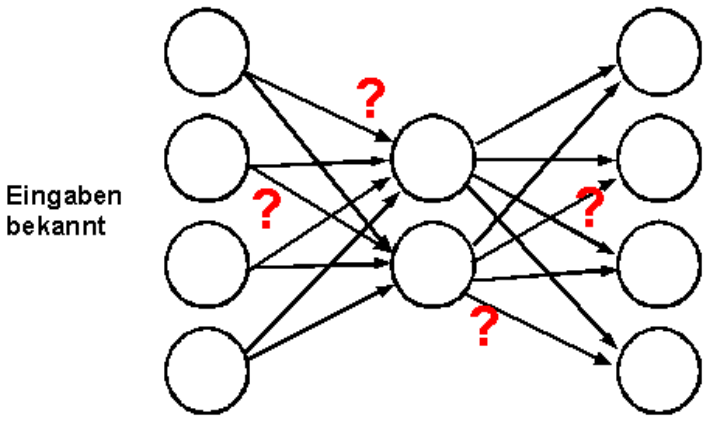


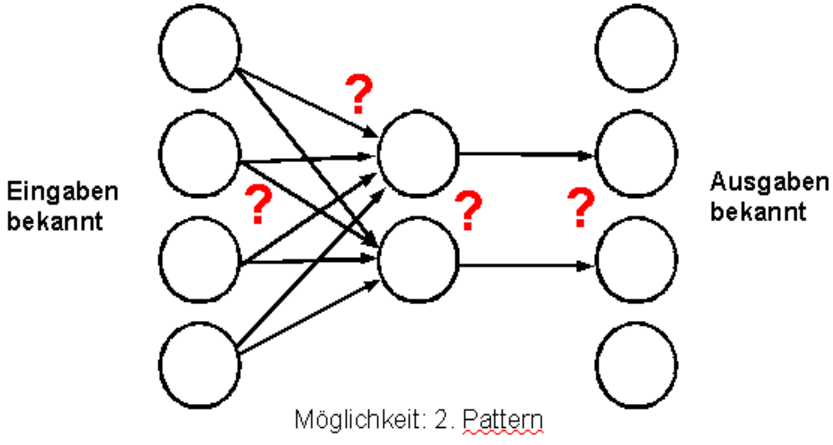
Schwierigkeiten (2)
Trainings-Software komplett in LabVIEW „geschrieben“



Schwierigkeiten (3)
Gradientenlernverfahren problematisch

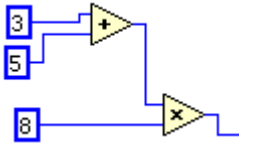
- Neuroboard ist eine „Blackbox“
  - Werte an versteckten Neuronen lassen sich nur mit Modell berechnen
    - wird ungenau
    - umständlich, da Modell sehr komplex





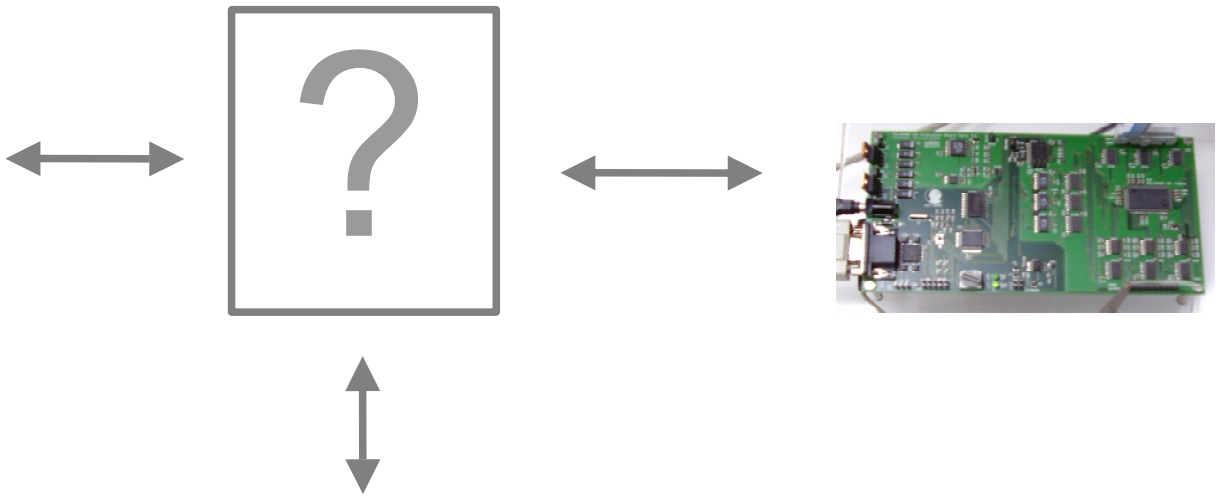
Schwierigkeiten (4)
Silimann Trainer/LabVIEW Source Code

- Source Code
  - wenig Kommentare/Dokumentation
  - Neuroboard-Befehle im Manual unvollständig
  - Kommunikationsprotokoll mehrfach implementiert
  - potenziell fehleranfällige Konstrukte:
    - z.B. Timer ohne Prüfung auf Überlauf
- LabVIEW (Datenflussorientiertes Programmierparadigma; meine Erfahrungen nach > 50 h)
  - Schätzung: 1-2 grafische Elemente pro Token in einer Programmiersprache
    - wird schnell komplex, insbesondere das „Verdrahten“
    - komplexe Algorithmen schwierig zu realisieren
  - Debuggen wenig komfortabel
    - stürzt manchmal ab, viel Klickarbeit, unerklärliche (?) Phänomene
  - kein Hinweis, das man zum Ausführen, die National Instruments VISA-Treiber benötigt
- Große Schaltungen konnte ich mitunter durch relativ wenige äquivalente C#-Anweisungen ersetzen (s.u.)




```
x=3+5*8;
```

Anspruchsvolle Integrationsaufgabe



Particle Swarm Optimization-Implementierung in C++

Implementierung (1)
LabVIEW-Versuch

- Naheliegende Lösung:
  - An Stellen, an denen im Source Code das Modell verwendet wird, Zugriffe auf das Neuroboard einfügen
- Probleme mit entsprechender Implementierung (~10 neue VIs):
  - Das Gradientenverfahren benötigt Eingangswerte der versteckten Neuronen (s.o.)
    - Lösung: Inverses Modell bilden (mit Hilfe von Intervallschachtelung)
  - Boardzugriff funktionierte nicht (mitunter KNN-Ausgabewerte von 29)
    - Ursache mit LabVIEW praktisch nicht zu finden
      - zumindest sehr zeitaufwendig

Implementierung (2)
NeuroBoard-API

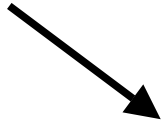
- NI VISA-Treiber auch direkt aus .NET verwendbar
- Versuch: mit C# auf Neuroboard zugreifen
  - klappte schnell und zuverlässig
- Implementierung einer NeuroBoard-API mit allen Board-Befehlen (auch den Undokumentierten...)
  - einfacher Zugriff auf die Neuroboard-Funktionen

7.11.1 readADC

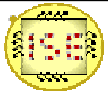
With this command you can read a single ADC channel. The bit width of the result is 12 bit. The channel number 0 is the temperature/AUX channel. The channels 1 to 10 are connected to the pins of the analogue input or the output, depending on the position of the input- and output switches.

Command:

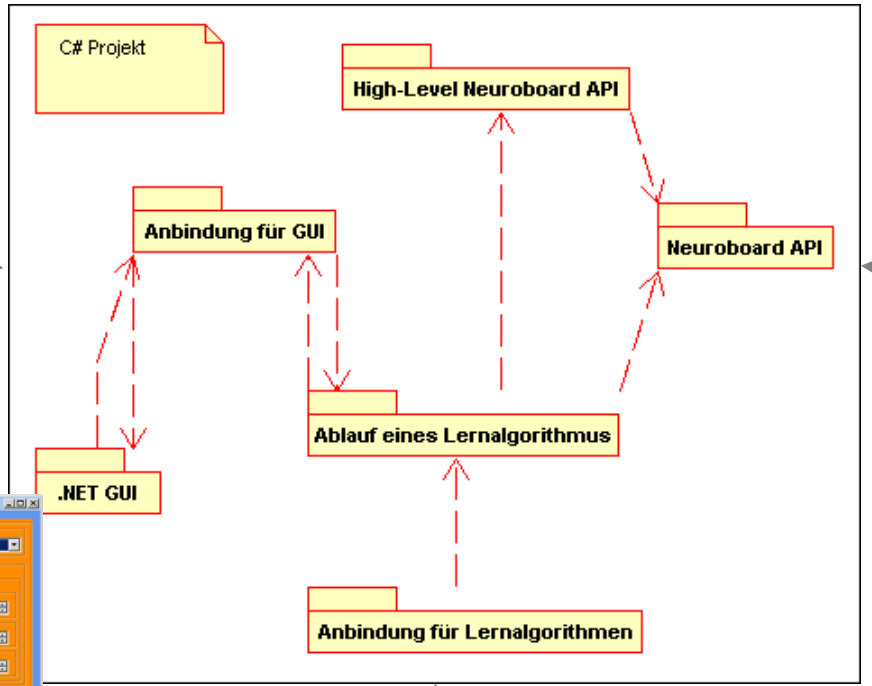
Code	Length	Payload	Checksum
0x24	0x01	ADC Channel	0xXX

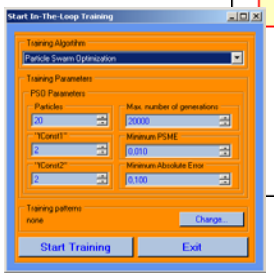


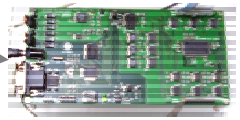
```
// readADC command (see 7.11.1)
public static Int16 ReadADC(byte ADCChannel)
```

Systemarchitektur







Particle Swarm Optimization-Implementierung in **C#**

Implementierung (4)
Features

- Modularer Aufbau
  - verschiedene Benutzerberflächen können angebunden werden
  - System kann um weitere Lernalgorithmen erweitert werden
- Programmcode hat recht hohes Qualitätsniveau
  - Implementierung robust
  - Code (einigermaßen) kompakt und übersichtlich
    - gut lesbar, wartbar und änderbar
  - Teile davon, in Drittanwendungen nutzbar (z.B. die „BoardAPI“)

Bewertung & Ausblick (1)

- Aufgabenstellung war meines Erachtens recht anspruchsvoll
  - viele unvorhergesehene Probleme
  - wenig Dokumentation
- In-the-loop-Learning sinnvoll im Praxis-Einsatz ?
  - Lernen dauert sehr lange (im Bereich von Stunden)
    - Einen Satz Gewichte hochzuladen, benötigt etwa 5 Sekunden
  - Flash-Speicher wird permanent überschrieben

Bewertung & Ausblick (2)

- Projekt evtl. für Silicann interessant
  - Implementierung von Trainings-Software in einer ganz anderen Technologie
    - Vergleich möglich, evtl. damit auch Verbesserung des eigenen SW-Entwicklungsprozesses (?)
    - Verwendung der .NET-Lösung brachte in meinem Fall eine Produktivitätssteigerung von geschätzten 500% (?)
      - (habe allerdings viel .NET & kaum LabView-Erfahrung)
  - mit der erstellten .NET-Infrastruktur sollte es mit relativ geringem Aufwand möglich sein, den kompletten Trainer nach C# zu portieren
    - könnte das machen, sofern Interesse besteht