# Hardwarezugriff und Signalanalyse mit Excel

# Messdatenerfassung:

Holen der Daten von einem Gerät, das über USB angeschlossen ist:  
Zugriff über die serielle Schnittstelle bzw USB (präsentiert sich auch als serielle Schnittstelle COMx)

Sehr einfach in .NET zb mit C# wenn man die Bibliothek System.IO.ports verwendet. Dort gibt’s ein SerialPort – Objekt, dass die Methoden open, close, read, write, dataexitsting usw kennt.

In C++ und in VB fertige Klassen bzw Objekte, die Funktionen der alten WinAPI (kernel32.dll) oder andere freie dlls verwenden.  
Es gibt auch

C++:   
Es gibt eine Klasse namens CSerial, mit der vollständiger Zugriff auf die Schnittstelle möglich ist.   
Man muss nur die COM Nummer und die Einstellungen kennen (Systemsteuerung / Gerätemanager)

Methoden der Klasse sind   
ModeSet … Übetragungsparameter einstellen  
SetDTR, ReseDTR, SetRTS, … die einzelnen Registerbits setzen und zurücksetzen  
Open, Close, SendData, ReadData, WriteCommByte, ReadCommByte  
Ablauf wie beim Schreiben und Lesen von Bytes.  
Üblicherweise sendet das Gerät in regelmäßigen Abständen und der Klient macht polling und schaut im Eingangsbuffer nach, ob schon was drinnen steht, dann liest er. Das Protokoll und die Zeiten muss man gut kennen bzw gut ausmachen, die serielle Schnittstelle ist aber vollduplex fähig, dh man kann in beide Richtungen gleichzeitig kommunizieren und muss nur aufpassen, dass die Variablenbuffer groß genug sind und rechtzeitig ausgelesen werden.

VisualBasic  
Man kann mit declare function auch auf die WinApi Funktionen (kernel32.dll) zugreifen. Oder die IO.dll oder Port.dll. Da gibt es einige, die sehr gut sind

Wenn man Funktionen (functions) oder Prozeduren (subs) verwendet

Schaut dann zB in vb so aus

Private **Declare Function** SetCommState Lib "**kernel32.dll**" (ByVal hCommDev As Long, ByRef lpDCB As DCB) As Long

oder

Private **Declare Sub** PortOut Lib "**IO.DLL**" (ByVal Port As Integer, ByVal Data As Byte)

Wenn man ein 64 Bit System hat wird’s schwieriger, da muss man die kernel32.dll „umbiegen“ oder gleich eine 64 Bit dll nehmen  
Gute Beispiel mit allen Funktionen die man braucht sind im Internet sind zu finden.  
Danach ist der Zugriff wieder einfach per read and write wie bei files

**Übernahme der Daten nach Excel:**Schreiben der Daten mit dem cells Befehl  
Wenn die Schnittstelle geöffnet ist

zB ein kB Daten einlesen und in die Spalte A schreiben  
  
Schnittstelle öffen  
COM1= com.open(„COM1“,9600, 1,1)

For i=1 to 1024  
 while COM1.read =““ ‘Warten auf Datenpaket  
 wend  
 COM1.read daten  
 cells(1,i)= daten  
next

Dann hat man alle Möglichkeiten von Excel zur Verfügung.

Wichtige Punkte:  
  
Application.ScreenUpdateing false setzen bevor Zellen manipuliert werden, sonst wird extrem langsam, bis zum Absturz , danach wieder true

VB Analysetoools aktiv setzen

## Fourieranalyse:

Aussage von Fourier: jedes stetige Signal lässt sich durch eine Überlagerung von regelmäßigen Schwingungen (zB Kosinus oder Sinus) mit steigender Frequenz darstellen.  
Die Fouriertransformation liefert Koeffizienten, die Informationen darüber liefern wie wichtig jeweils der Beitrag eines Kosinus zum Erscheinungsbild des Signales ist.  
Die Koeffizienten werden der Reihe nach durchnummeriert: Koeffizienten mit niedrigen Nummern gehören zu langwelligen (niederfrequenten) Mustern im Signal also zum groben Erscheinungsbild des Signals, die hohen zu den kurzen Mustern, also den kleinen Strukturen und Details.   
Man erhält genau so viele Koeffizienten wie die Messreihe Werte hat. Aus den Koeffizienten kann die ursprüngliche Reihe durch eine inverse Fouriertransformation wieder genau hergestellt werden.  
Wenn man bei der Wiederherstellung Koeffizienten mit hohen Nummern auf 0 setzt, also weglässt, erhält man eine Messreihe in der die Details fehlen der grobe Verlauf aber gut abgebildet wird.  
Das macht man zB bei mp3s, mpegs und jpegs so. Man speichert nicht alle Koeffizienten sondern nur die niedrigen, dadurch spart man erheblich Speicherplatz und bekommt doch noch ein Signal, das das Original gut wiederspeigelt (mpegs brauchen nur ein 10.tel des Speicherplatzes der vollen WAV Datei und klingen nur ein bißchen „stumpfer“, bei jpegs und mpges fehlen Details, sie sind etwas unschärfer)  
Die Fast Fourier Transformation (FFT) ist eine besonders schnelle Version, funktioniert aber nur mit einer Anzahl vom Messdaten, die eine Potenz von 2 sind.

## Vorgangsweise bei der FFT

Das Signal (eine Datenreihe) auf eine Potenz von 2 mit 0en Auffüllen  
FFT machen  
Ergebnis sind komplexe Zahlen, die symmetrisch um die Mitte gespiegelt sind.  
Am Anfang (und am Ende wegen der Spiegelung) sind die tiefen Frequenzen, also die langwelligen Anteile des Signals, um die Mitte herum die hochfrequenten (kurzen) Signaländerungen.  
Man kann das Signal zB glätten in dem man nur die Wert am Anfang und am Ende (gleich viele) übrig lässt und die restlichen rund um die Mitte (also die hohen Frequenzen) auf 0 setzt.

Dann Inverse FFT und man erhält ein gefiltertes Signal  
Sowas wird auch bei jpeg und mpeg gemacht, man muss die auf 0 gesetzten Werte nicht speichern und erhält eine Datenreduktion, dafür geht Detailschärfe verloren

Grafik für Tiefpass, Hochpass und Bandpass

Kleiner Trick:

was macht man wenn man die Funktionen und ihre Argumente nicht kennt?  
Im Internet herumsuchen – oder viel einfacher (wenn man weiß was man tut):  
Makro aufzeichnen und die Arbeitsschritte händisch durchführen -> Das Makro enthält dann die notwendigen Befehle mit allen Parametereinstellungen.