

# CDIO opgave forår 2014 - Datakommunikation Del 2

*Alternativt projekt*

*Dataopsamling fra tests af piezoelektriske krystaller.*

Morten Hesselbjerg, s017704

Christian Budtz, s134000

Rikke Christina Hansen, s120359

Rúni Egholm Vørmadal, s134004

Martin Nielsen, s123064

Eirik Oterholm Nielsen, s123006

Magnus Brandt Sløgedal, s103185

# 51\_02324\_D2\_timeskema

Dato	Navn	Design	Impl.	Test	Dok.	Andet	I alt	Bemærk
24/03/2014	Christian	3					3	UML
31/03/2014	Christian		4				4	JDBC
31/03/2014	Martin		4					
01/04/2014	Christian		4				4	DB
01/04/2014	Martin		3					
02/04/2014	Christian		3				3	
07/04/2014	Christian		7	1			8	
07/04/2014	Rúni		7	1			8	
07/04/2014	Morten		7					
08/04/2014	Christian		3				3	
08/04/2014	Eirik		6	3			9	
08/04/2014	Martin		3			3		
14/04/2014	Rúni		7	2		1	10	
15/04/2014	Christian		5		0,5		5,5	DB
15/04/2014	Rúni		9	3			12	
15/04/2014	Martin				1			
16/04/2014	Rúni		4	1		1	6	
16/04/2014	Eirik		1	1	1		3	
17/04/2014	Rúni		4	1			5	
17/04/2014	Morten		1	1				
18/04/2014	Rúni		5	1		1	7	
18/04/2014	Eirik		3				3	
19/04/2014	Rúni		2	1	1		4	
20/04/2014	Christian			0,5	2	0,5	3	
20/04/2014	Rúni		3	2	1		6	
20/04/2014	Eirik		1	0	1		2	
20/04/2014	Morten		1	1				
21/04/2014	Rúni		1	1	4		6	
21/04/2014	Christian				2		2	
21/04/2014	Martin				2			
21/04/2014	Eirik				1		1	
21/04/2014	Magnus			2	3		5	
21/04/2014	Morten		6	1				
22/04/2014	Rúni						0	
22/04/2014	Christian				3		3	
22/04/2014	Morten		3		1			
22/04/2014	Magnus		3					
	Total						125,5	
	Christian	3	26	1,5	7,5	0,5	38,5	
	Eirik	0	11	4	3	0	18	
	Martin	0	10	0	3	3	16	
	Morten	0	18	3	1	0	22	
	Rikke	0	0	0	0	0	0	
	Rúni	0	42	13	6	3	64	
	Magnus	0	3	2	3	0	8	
							166,5	

[Indledning](#)

[Status](#)

[Beskrivelse af programflow](#)

[Design](#)

[CDIO Demo.exe](#)

[DasyFileReader](#)

[CConnector](#)

[DataBaseController](#)

[Begrænsninger](#)

[Udviklingsplan](#)

[Sidste iteration](#)

[Bilag](#)

[Protocol for java to C# socket communication.](#)

[Screenshots](#)

## Indledning

Jævnfør aftalen har vi arbejdet på vores alternative projekt og forsøgt at implementere features, der modsvarer de stillede opgaver i vægtprojektet. Særligt har vi forsøgt at arbejde på vores TCP/IP datakommunikation mellem en C# komponent og vores hoved-Java program og forsøgt at sikre robust kommunikation og mulighed for at genoptage arbejdet efter uventet afslutning af dataopsamling. Vi anvender i vores løsning en SQL-server og gemmer kontinuert data i databasen, således at data ikke går tabt undervejs. Lukker man programmet, eller går det ned, kan man loade sin måleserie og fortsætte fra sidste succesfulde måling.

## Status

Vi har nu udviklet det grundlæggende programflow i projektet. Data kan opsamles fra to forskellige kilder, hhv. fra et USB-device tilsluttet computeren og fra DasyLab filer genereret af kundens egen software. USB-modulet kan opsamle analoge data fra en tilsluttet mikrometerskrue. Selve tilslutningsprogrammet til USB device er udviklet i C#, da der kun findes drivere til dette. Når C# programmet kører, opstarter man et Java-program og kommunikation med C# programmet foregår så over en TCP/IP forbindelse via vores egen protokol.

Al data gemmes for hver måling i DTU's mySQL database og det er muligt at genoptage en måleserie (batch), der er blevet afbrudt før tid.

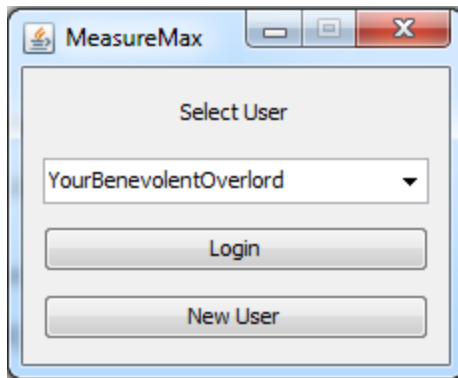
## Beskrivelse af programflow

For at få et indtryk af hvad vores projekt aktuelt kan, er det nødvendigt at starte vores C# server, der er konstrueret til at opsamle data fra det tilkoblede USB-device. Denne startes fra filen CDIO\_Demo.exe, placeret i projektmappen C#Code\CDIO\_Demo uden mccdqv v2.zip.

C# -serveren opretter en TCP/IP serversocket og afventer en forbindelse. Bruger-interaktion og al anden logik ligger i vores java-program. Den vedlagte udgave af C# programmet er en testudgave, der kan returnere tilfældige værdier, når man ikke har tilsluttet det relevante måleapparat. Vores produktionsudgave af C# serveren, nødvendiggør installationen af et 256 MB stort driverprogram, så til test og opgavebedømmelse har vi udviklet vores letvægtsudgave.

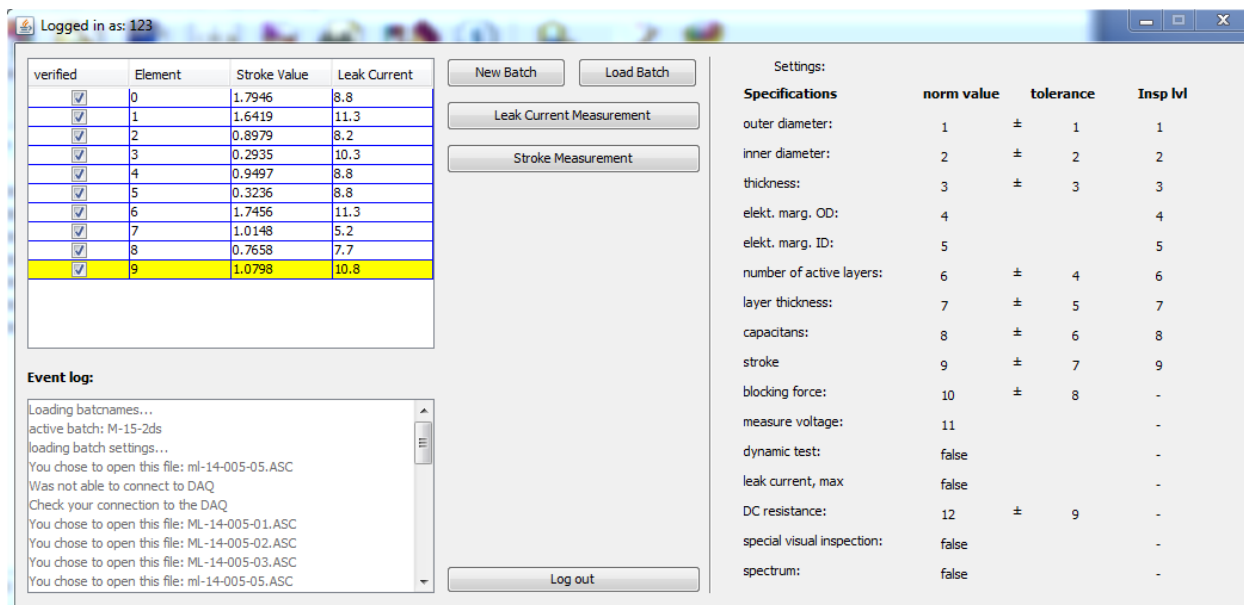
Vores javaprogram startes fra default package fra MeasureMain klassen.

Kører man programmet som det er nu, gives man først muligheden for at logge ind med sit brugernavn, eksisterer brugeren ikke kan man oprette en ny bruger. Brugere er gemt i mySQL-databasen. Vi har endnu ikke gjort det muligt at slette en bruger.



*Figur 1: Loginscreen* Når man logger ind kan man enten vælge en eksisterende bruger, eller skrive et navn i tekstfeltet og oprette en ny bruger. Brugerlisten indhentes fra SQL-databasen.

Herefter kan man oprette eller load et batch fra databasen og lave målinger tilhørende det batch. Hvert batch er associeret med en måleprofil, bestående af et antal settings der ligeledes gemmes i databasen. Målinger kan ikke udføres uden af være knyttet til et batch og en tilhørende måleprofil, men det er ikke nødvendigt at indtaste noget i profilen - man kan altså starte målinger uden at have indtastet settings og disse gemmes blot blanke.



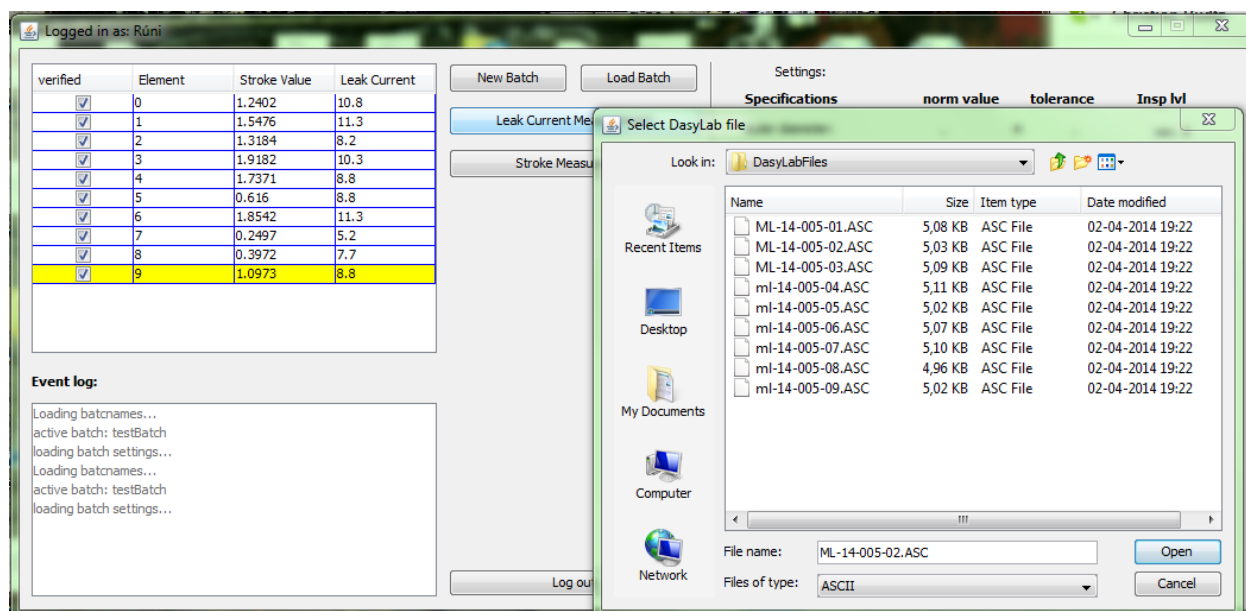
*Figur 2: Main Menu.* Når man er logget ind kommer man til Main Menuen. Man kan vælge imellem at load en eksisterende batch eller at oprette en ny batch. Hvis man vil lave en måling skal man vælge en ASC-fil, som så optræder som et element på listen.

Specifications	norm value	tolerance	inspection level
outer diameter:	<input type="text"/> ±	<input type="text"/>	<input type="text"/> min. 5
inner diameter:	<input type="text"/> ±	<input type="text"/>	<input type="text"/> min. 5
thickness:	<input type="text"/> ±	<input type="text"/>	<input type="text"/> min. 5
elekt. marg. OD:	<input type="text"/>		<input type="text"/>
elekt. marg. ID:	<input type="text"/>		<input type="text"/>
number of active layers:	<input type="text"/> ±	<input type="text"/>	<input type="text"/>
layer thickness:	<input type="text"/> ±	<input type="text"/>	<input type="text"/>
capacitans:	<input type="text"/> ±	<input type="text"/>	<input type="text"/> min. 10
stroke:	<input type="text"/> ±	<input type="text"/>	<input type="text"/> AQL 0.65
blocking force:	<input type="text"/> ±	<input type="text"/>	<input type="text"/>
measure voltage:	<input type="text"/>		<input type="text"/>
dynamic test:	<input type="checkbox"/>		<input type="text"/>
leak current, max:	<input type="checkbox"/>		<input type="text"/> min. 20
DC resistance:	<input type="text"/> ±	<input type="text"/>	<input type="text"/>
special visual inspection:	<input type="checkbox"/>		<input type="text"/> AQL 0.65
spectrum:	<input type="checkbox"/>		<input type="text"/>

*Figur 3: New Batch Menu.* Her indtaster man specifikationer for et batch. Man kan også loade gamle batch specifikationer, hvis de er blevet gemt først.

Når batch og måleprofil er fremsøgt, kan målingerne starte. Er der allerede foretaget målinger på den givne batch (eks. ved tidligere login), loades disse og man kan fortsætte målearbejdet. Målingerne bliver så snart de er taget, gemt i databasen.

Måling af lækstrøm (Leak current) foregår ved indlæsning af en fil, som man vælger i en dialogboks (Figur 4). Vi har gjort, at det er kun muligt at hente ascii/asc filer, da det er dette format kunden bruger, og så har vi for brugervenlighed gemt stien til sidste mappe man loadede en fil fra, så man ikke skal browse så langt næste gang. Dette gemmes i en property fil. Når programmet afprøves kan test ASC filer findes i mappen DasyLabFiles i projektmappen. Når filen er valgt, parses denne og data fra filen omsættes til en værdi, der straks gemmes i databasen og vises i målevinduet i øvre ve. hjørne.



Figur 4. Indhentning af data fra DasyLab filer til Leak Current Measurement. Data parses fra filer genereret af DasyLabProgramet.

Stroke målinger foretages direkte ved tryk på Stroke Measurement knappen. Målingen gemmes med det samme i databasen og vises i målevinduet.

Det er muligt at navngive en måleprofil i databasen og senere hente den frem igen - således at hvis det er indstillinger man bruger ofte, så slipper man for at taste dem alle ind hver gang de skal bruges.

Vores hovedvindue indeholder en log (se evt. Figur2, kan ses nede i venstre hjørne), der udskrifter hvad der aktuelt foregår, mht. kommunikation med C# del og indlæsning af DasyLabFiler. Når kommunikationen foregår, udskrives dette til brugeren, og opstår der fejl, udskrives dette ligeledes her. Er C# serveren eks. ikke opstartet eller fejler forbindelsen, gives en besked her - ligesom andre fejl i dataopsamlingen meldes her - således at brugeren kan korrigere problemer. Aktuelt er brugerens valgmuligheder begrænsede - i de næste iterationer planlægger vi at udvide brugergrænsefladen til at understøtte en mere fleksibel arbejdsgang. Som nævnt har vi - for at vi/underviser/hjælpelærere skal kunne teste og demonstrere at forbindelsen virker - i projektet gjort sådan at C# komponenten sender tilfældige værdier mellem 0,5 og 2, da vi ikke har USB devicen koblet til.

## Design

For at håndtere en robust TCP/IP forbindelse har vi skrevet en java klasse - CConnector, der opretter en TCP/IP Socket og skaber forbindelse til vores C# komponent. Aktuelt returnerer C#-programmet dummydata, da vi ikke har fået fat i den mikrometerskrue som vores kunde bruger. Vi har lavet vores egen protokol til at håndtere fejl i kommunikationen, og har identificeret 5 mulige fejlmeddelelser som kan ses under bilag med titlen "Protocol for java to C# socket communication".

For at sikre robusthed i tilfælde af nedbrud eller præmatur afslutning af måleserie har vi udviklet en løsning til at hente og gemme data i en database. Vi bruger aktuelt en forbindelse til den mySQL database vi har fået stillet til rådighed i Diskret matematik og databaser. For at få fremdrift i projektet har vi kun en overordnet exception (DatabaseException) som bliver kastet hvis noget går galt - enten med SQL-syntaks eller med forbindelsen til serveren. Dette vil blive subclasset senere for en mere robust fejlhåndtering. En praktisk ekstra feature, kunne være muligheden for at gemme data lokalt, indtil database forbindelsen igen kan oprettes.

## CDIO\_Demo.exe

C# Serveren der afventer forbindelse fra Java programmet- Når connection oprettes og der modtages en besked over TCP/IP enten aflæses målinger fra USB device't eller returneres test data.

Vi har følgende 3 væsentlige klasser der sørger for data input/output.

## DasyFileReader

DasyFileReader er den klasse, der læser filer fra DASYLab, gemt i .ASC format. Den parser ASC-filen og udlæser de measurements som er gemt i filen. Herefter kan de relevante measurements udvælges og gemmes i databasen.

## CConnector

CConnector er den klasse, der taler mellem vores java og C# del af projektet. Vi har lavet en forbindelse, og flyttet data over. I øjeblikket laver vi en request, og får data'en tilbage i en string. Vi håndterer også flere fejl, som fx. der er ingen forbindelse. Den fulde protokol for kommunikation mellem Java og C# delen er vedlagt som bilag.

For at begynde, skal man bruge setIp metoden. Så skal man bruge readMeasurments metoden. Dette vil sende en request til C# delen, der kigger på DAQ'en, og returnerer measurements. Når man er færdig, termineres forbindelsen med disconnect metoden. enten give dig værdierne der ligger på DAQ'en, eller returnere en fejlmeddelelse, som fx "DAQ not connected or malfunctioning".



## **DataBaseController**

DataBaseController er klassen der henter ting fra vores SQL database. Databasen indeholder alt fra brugere, til programindstillinger, til testdata. DatabaseControlleren omsætter forespørgsler fra andre programmer til SQL-queries. Queries eksekveres via SQLConnector klassen.

## **Begrænsninger**

C# programmet skal aktuelt startes manuelt - for brugervenlighedens skyld planlægger vi at det skal startes automatisk. Log-vinduet er ikke up to date, og der kommer intermitterende meddelelser, som ikke stemmer overens med fejltilstanden. Målingerne skal foretages konsekutivt og det er aktuelt ikke muligt at kassere enkeltmålinger.

## **Udviklingsplan**

Vi har fulgt vores plan nogenlunde, og derfor ser vores plan for sidste iteration stort set ud som den gjorde i starten. Vores GUI mangler stadig knapper og labels til de resterende funktioner, der også skal implementeres. Funktionalitet, som vi i dialog med Noliac har tilføjet projektbeskrivelsen, er indsat i planen. Vi har en HTML Mockup, udviklet til at understøtte dialogen med Noliac, men der er ingen reel funktionalitet i denne endnu.

## **Sidste iteration**

Bugfixing, optimering og evt. udvidelse af implementering

1. Kravspecificering
  - a. Løbende tæt dialog med Noliac, mhp. validering af løsningsforslag og prioritering af udviklingsopgaver.
2. Design
  - a. Optimering af kodebase/GUI
    - i. Test af grænsetilfælde
    - ii. Inputvalidering
    - iii. Bedre exception handling
    - iv. Bedre/mere robust håndtering af SQL-connection
3. Implementering
  - a. Manglende funktionalitet i Javaprogram
    - i. Slet bruger(e)
    - ii. Superbruger funktionalitet
    - iii. Måleprofil redigering
    - iv. Slette/redigere gemte profiler
    - v. Verifikation af måledata - mulighed for at kassere fejlagtige måledata
    - vi. Fleksibilitet af måleflow - så man evt. kan bestemme hvilket element, der bliver målt på
    - vii. Program setup - mulighed for at konfigurere eks IP-adresser, måleporte, standard profiler.
    - viii. Gemt data, samles i én property fil (nu gemmes data i flere properties)

- b. Implementering af SQL/JSP i HTML/CSS grænseflade
      - i. Grundlæggende funktionalitet til at udhente de indsamlede måledata.
- 4. Test.
  - a. Unit testing af essentielle databehandlingsklasser
- 5. Evt:
  - a. Implementering af 'custom' rapportgenerering.
  - b. Måske rapportopslag for kundens kunder efter batchnr?
  - c. Statistisk udregning af sample size for at garantere en vis tolerance med en given sandsynlighed.
  - d. Brugerstyring
  - e. Større robusthed af database-interface
    - i. evt. offline mode og synkroniserings funktionalitet (lav prioritet).

Mål: Optimering af løsning, evt. implementering af flere use cases. Kunden skal kunne Bruge programmet i drift

Slut: 11/5

## Bilag

### **Protocol for java to C# socket communication.**

The C# component should be set to listen, and java will connect when the program starts. By default, java will try to connect to localhost port 4567, but this can be overloaded.

When the connection has been established, the following requests and responses are allowed.

Request 1:

Java sends a string to C# in the following format: port+";"+number+";"+period+";<EOF>" where port is the desired port on the DAQ to read from, number is the desired amount of readings, and period is the time between readings  
port, number and period are ints.

Response 1:

C# then returns a string in the format: data+";"+timeStamp+";"  
this is repeated number times.

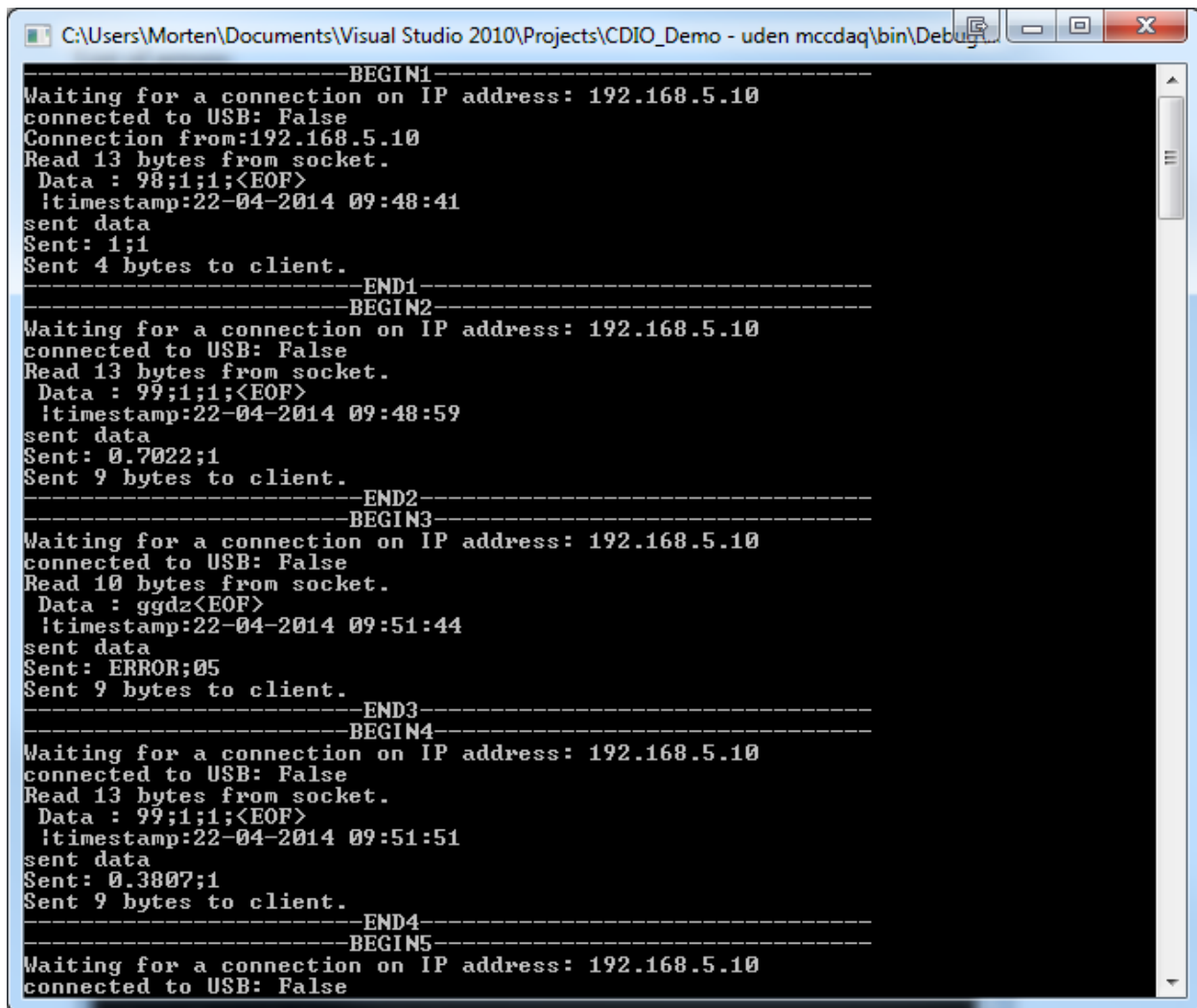
Data is the reading, a float value, and timestamp is the time since the previous reading(long).

If an error has occurred, C# returns a string in the format: ERROR+";"+number+";"  
error is a string that reads error, and number is an int, that describes what error has occurred.

List of errors:

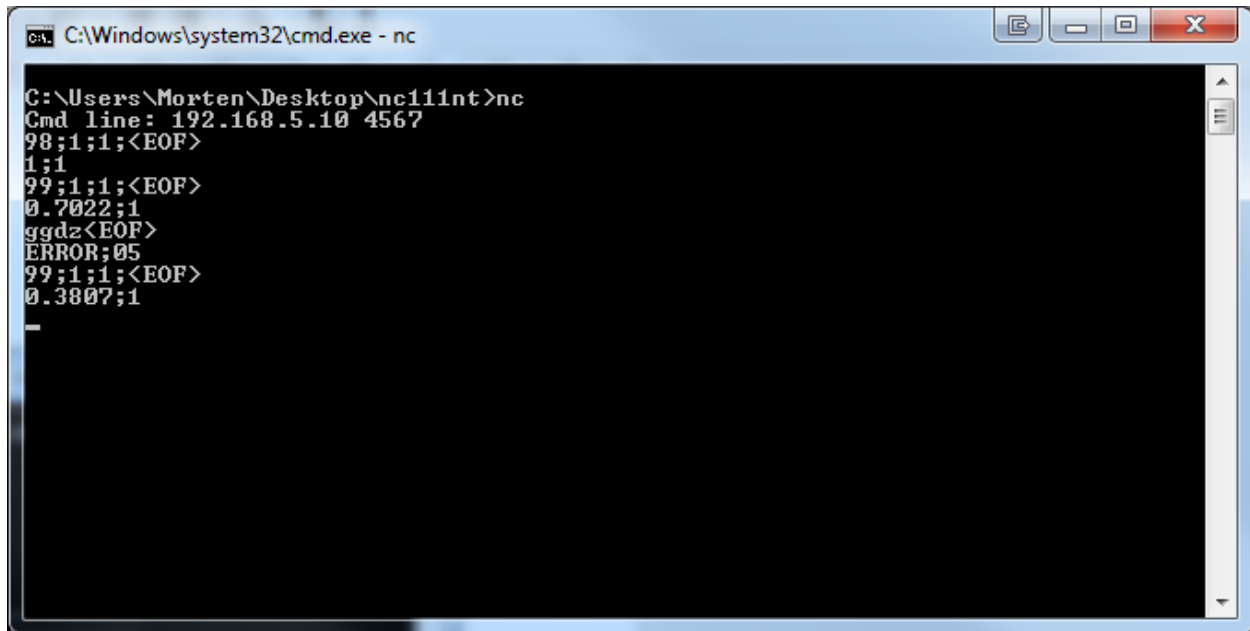
- 01: DAQ not connected or malfunctioning
- 02: Illegal port
- 03: Illegal number of measurements
- 04: Illegal period
- 05: Message does not conform to protocol

## Screenshots



```
C:\Users\Morten\Documents\Visual Studio 2010\Projects\CDIO_Demo - uden mccdaq\bin\Debug...
-----BEGIN1-----
Waiting for a connection on IP address: 192.168.5.10
connected to USB: False
Connection from:192.168.5.10
Read 13 bytes from socket.
Data : 98;1;1;<EOF>
!timestamp:22-04-2014 09:48:41
sent data
Sent: 1;1
Sent 4 bytes to client.
-----END1-----
-----BEGIN2-----
Waiting for a connection on IP address: 192.168.5.10
connected to USB: False
Read 13 bytes from socket.
Data : 99;1;1;<EOF>
!timestamp:22-04-2014 09:48:59
sent data
Sent: 0.7022;1
Sent 9 bytes to client.
-----END2-----
-----BEGIN3-----
Waiting for a connection on IP address: 192.168.5.10
connected to USB: False
Read 10 bytes from socket.
Data : ggdz<EOF>
!timestamp:22-04-2014 09:51:44
sent data
Sent: ERROR;05
Sent 9 bytes to client.
-----END3-----
-----BEGIN4-----
Waiting for a connection on IP address: 192.168.5.10
connected to USB: False
Read 13 bytes from socket.
Data : 99;1;1;<EOF>
!timestamp:22-04-2014 09:51:51
sent data
Sent: 0.3807;1
Sent 9 bytes to client.
-----END4-----
-----BEGIN5-----
Waiting for a connection on IP address: 192.168.5.10
connected to USB: False
```

Bilag 5: C# komponent. Mellem linjerne Begin3 og End3 vises eksempel på fejlkommunikation



A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\system32\cmd.exe - nc". The command prompt shows the following text:

```
C:\Users\Morten\Desktop\nc111nt>nc
Cmd line: 192.168.5.10 4567
98;1;1;<EOF>
1;1
99;1;1;<EOF>
0.7022;1
ggdz<EOF>
ERROR;05
99;1;1;<EOF>
0.3807;1
-
```

*Bilag 6: Screenshot af kommunikation sendt til C# komponent (via Netcat)*