CRANN

NEW CONTROL SOFTWARE FOR CERBERUS 3D NANOINDENTATION SYSTEM

by

Bhakt Vatsal Trivedi
(2010041)

SUPERVISORS:

Mr. Saket Sourav
Research Engineer
IIITDM - Jabalpur

Dr. Graham L. W. Cross
PI, CRANN Nanotechnology Institute
Trinity College Dublin

Computer Science and Engineering Discipline

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DESIGN AND MANUFACTURING, JABALPUR
(End Term Report - 2013)

Acknowledgement

I want to thank my **Prof. Graham Cross, FTCD, CRANN, Trinity College Dublin** for the great opportunity to work in this project. It was not possible to attain anything in this project without his guidance and support.

I also want to thank **Mr. Saket Sourav** for his crucial inputs on the project. His valuable suggestions helped me a lot in the initial phase of the project and also throughout the project.

I also want to thank **Dr. Johann De Silva(TCD)** and **Dr. Mithun Chowdhury(TCD**) for listening to my small doubts and keeping patience while solving them.

At last I want to thank my institute, IIITDM-J, and internship board for giving me this golden opportunity to explore and test our knowledge.

(Bhakt Vatsal Trivedi)

17-11-2013

About CRANN

CRANN (Centre for **R**esearch on **A**daptive **N**anostructures and **N**anodevices) is a Science Foundation Ireland (SFI) funded Centre for Science, Engineering and Technology (CSET). It is based in Trinity College Dublin (TCD), is the college's largest research institute, and works in partnership with University College Cork (UCC).

CRANN works at the frontiers of nanoscience developing new knowledge of nanoscale materials, with a particular focus on new device and sensor technologies for the ICT, biotechnology and medical technology sectors, with a growing interest in energy related research.

CRANN combines world-class fundamental and applied research activity with a vibrant culture of industry engagement and commercialization. We have a strong track record of successful collaboration and partnership with industry and of licensing, in conjunction with the technology transfer offices of TCD and UCC, intellectual property developed in CRANN to companies for commercial application.

Cross Nanomechanics Group

Cross nanomechanics group works in CRANN with Prof. Graham L.W. Cross as the Principal Investigator of the group. At Cross group, the main areas of research are Nanoindentation and Nano-Lithography. The Cross group owns some of the finest instruments for the analysis of the samples at nano level.

CRANN

Certificate

This is to certify that the project report entitled, **New Control software for Cerberus 3-D Nano indentation system**, submitted by **Bhakt Vatsal Trivedi** in partial fulfillment of the requirements for the completion of PBI Project in Computer Science and Engineering at the PDPM Indian Institute of Information Technology, Design and Manufacturing Jabalpur is an authentic work carried out under my supervision as well as guidance and this work has not been submitted elsewhere.

Date:- 18/11/2013

Place:- Dublin, Ireland

[illegible]

Graham L. W. Cross
CRANN PI and Asst. Professor, TCD School of Physics

Index

Chapter 1. Introduction

1.1 The Project

The purpose of this project was to upgrade the existing control software system that runs the "Cerberus". The goals of the project were as follows:

To replace the HT Basic GUI elements with a modern GUI environment running under Windows 7 and written in a modern (e.g. scripted, e.g. PyLab (Python), Matlab, Igor Pro, or other) language suitable for scientific programming and real-time control, data storage and data display and review.

To encapsulate the existing HT Basic intermediate and low level subroutines in the new scripting language to make an API for easy extension of high level GUI code in the future.

To implement the project with minimal interruption to the use of Cerberus. This will be done by writing a Cerberus hardware simulator that will essentially be a separate program run on a separate PC or laptop, which will receive and return simulated electronic signals through a DAQ card.

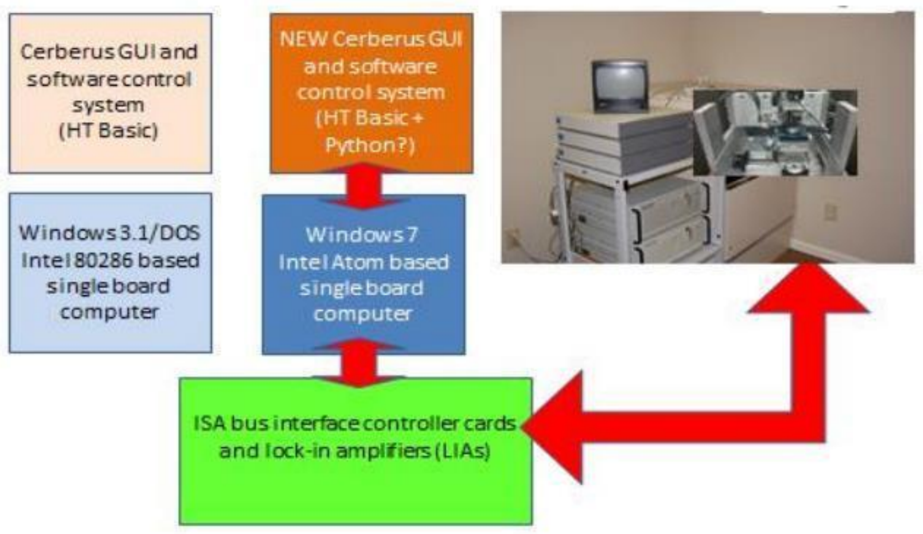

Cerberus GUI and software control system (HT Basic)
NEW Cerberus GUI and software control system (HT Basic + Python?)
Windows 3.1/DOS Intel 80286 based single board computer
Windows 7 Intel Atom based single board computer
ISA bus interface controller cards and lock-in amplifiers (LIAs)

Figure 1.1: The whole idea of the project.

Now let us go in the details of the Nanoindentation process and the machine.

1.1.1 Nanoindentation

Evaluation of mechanical properties of thin films has become important with the extending application of functional thin films, such as the protection film in the magnetic recording media and heads, the interconnects in the integrated circuit, the surface

modified layer in the ion implanted materials, etc. Nanoindentation is a powerful and simple means for evaluating the mechanical properties of thin films. The data of nanoindentation measurement is usually expressed by a curve of applied load fa versus displacement (penetration depth) h.
In nanoindentation small loads and tip sizes are used, so the indentation area may only be a few square micrometres or even nanometres. This presents problems in determining the hardness, as the contact area is not easily found. Atomic force microscopy or scanning electron microscopy techniques may be utilized to image the indentation, but can be quite cumbersome. Instead, an indenter with a geometry known to high precision (usually a Berkovich tip, which has a three-sided pyramid geometry) is employed.
During the course of the instrumented indentation process, a record of the *depth* of penetration is made, and then the area of the indent is determined using the known geometry of the indentation tip. While indenting, various parameters such as load and depth of penetration can be measured. A record of these values can be plotted on a graph to create a *load-displacement curve* (such as the one shown in Figure 1.2). These curves can be used to extract mechanical properties of the material.[2]

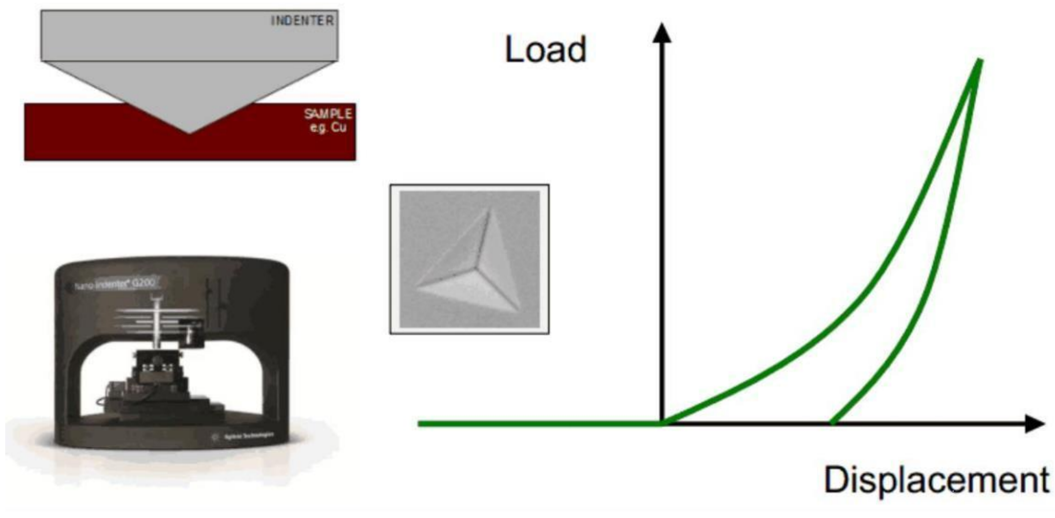


Figure 1.2: Indentation Technique and plot for the result.

1.1.2 Cerberus Nanoindentation System

Recently, several nanoindentation instruments were commercialized. **Cerberus is one such instrument, which was developed by CRANN , in 1990's**. Cerberus is controlled by an electronic hardware system that consists of an ISA-bus backplane which holds a number of ISA daughter-boards that control various functions of the indenter. These include interface cards to three external lock-in amplifier (LIA) boxes that are used to run the dynamic mode of the instrument in each of the x, y, and z axes. The entire system is managed by a Single Board Computer (SCB) ISA board, which is currently an old Intel 80286 based PC running Microsoft Windows 3.1 (i.e. DOS)[3].

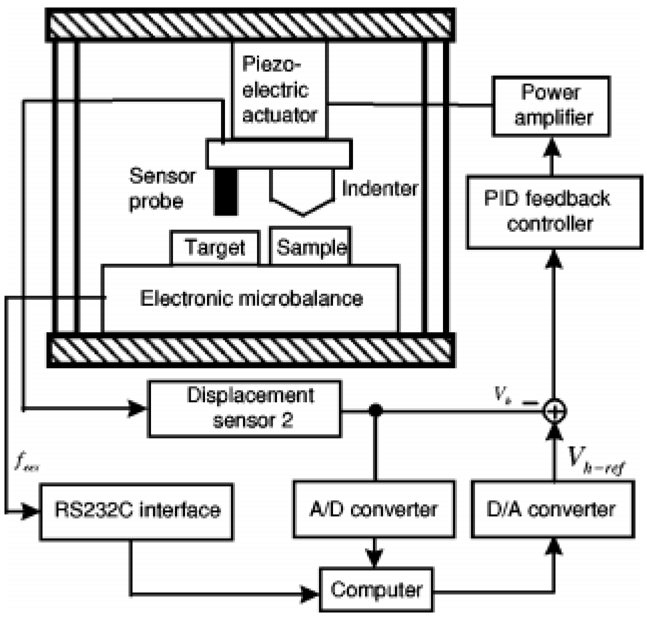


Figure 1.3: Block diagram for Cerberus

The **Cerberus controller software** is a large program written in the HT Basic interpreted language. It consists of low-level subroutines that communicate with the ISA boards and LIAs, intermediate level algorithms that make Cerberus perform nanoindentation tests by making calls to the low-level subroutines, and high level GUI elements that interact with the user.

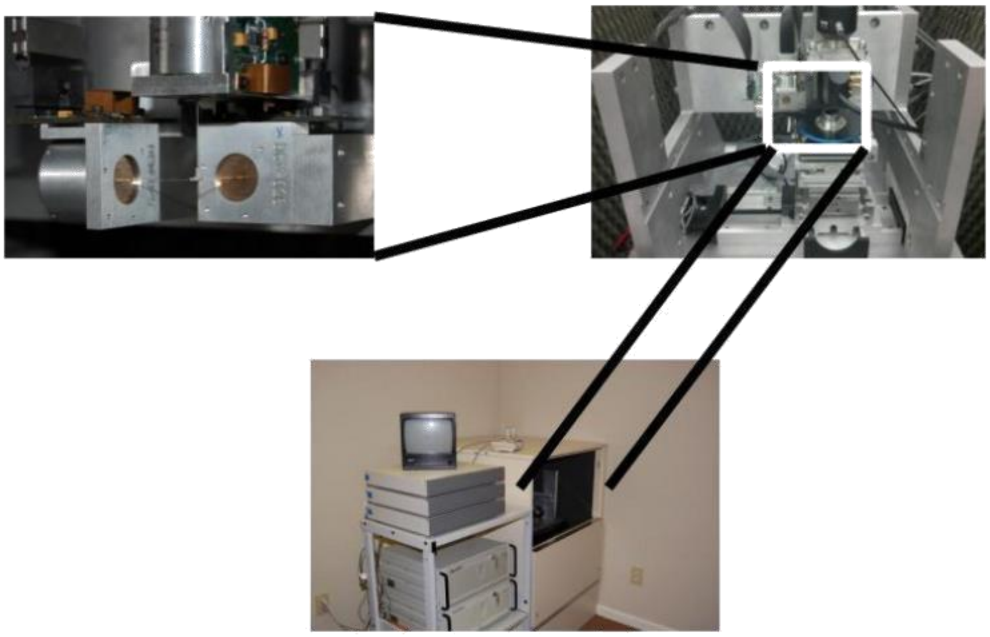


Figure 1.4: Set up for Cerberus

1.2 Time Line for the project

After deep discussion and analysis of the universe of discourse, we ended up with the following time line for the project:

Task	June	July	Aug.	Sept.	Oct.	Nov.
Assess requirements, decide and order software and hardware, start on GUI elements for user inputs, live display of data, etc.	🟦					
Write and test the Cerberus hardware simulator		🟦	🟦			
Write and test the new Cerberus software control system using the simulator			🟦	🟦		
Test new software system for the real Cerberus					🟦	
Documentation and report write up						🟦

Table 1.1: Time line of the project

1.3 Concepts after initial discussions

1.3.1 Concept 1

In this concept, we came out with the following ideas:

- We will connect a new DAQ card to the old system which will read the experiment data from Cerberus hardware.
- There will another DAQ card connected to the modern Win7 machine and also connected to first DAQ card via external connection medium.
- The first DAQ card will buffer out the experiment data to the second DAQ.
- The GUI system running on the modern Win7 system will read this data and do appropriate processings.

This concept is depicted in the following figure:

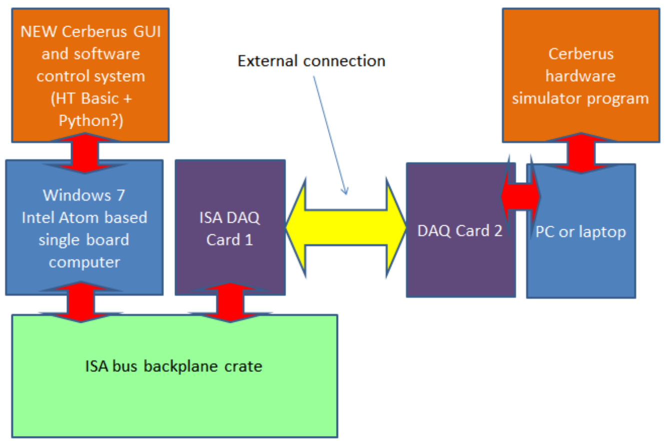

NEW Cerberus GUI and software control system (HT Basic + Python?)
External connection
Cerberus hardware simulator program
Windows 7 Intel Atom based single board computer
ISA DAQ Card 1
DAQ Card 2
PC or laptop
ISA bus backplane crate

Figure 1.5: Concept 1

1.3.2 Concept 2

In this concept a different approach was adopted. The following figure illustrates the contrast between the two strategies. In the second plan, we came out with the idea to transfer the experiment data between the two machines using some kind of interface like USB, RS-232, Networked Drive etc. The figure illustrates that the first concept might be too much considering the time complexity of the project.

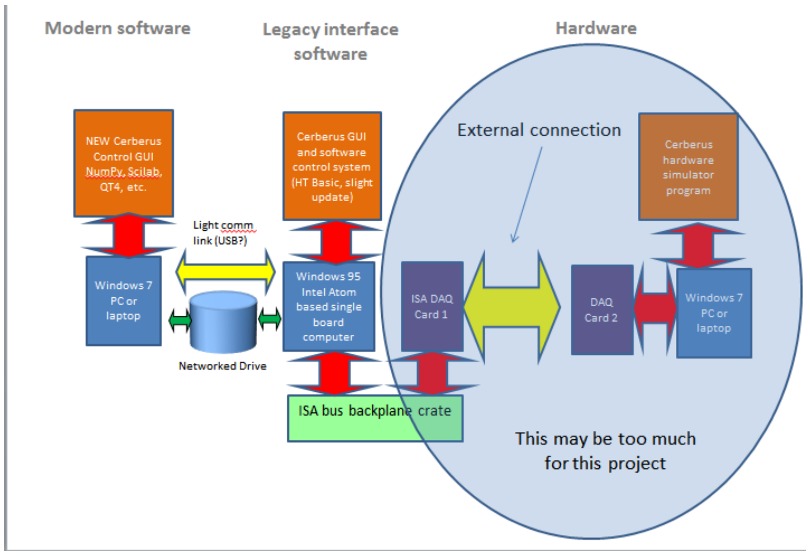

Modern software
Legacy interface software
Hardware
NEW Cerberus Control GUI NumPy, Scilab, QT4, etc.
Cerberus GUI and software control system (HT Basic, slight update)
External connection
Cerberus hardware simulator program
Light comm link (USB?)
Windows 7 PC or laptop
Windows 95 Intel Atom based single board computer
ISA DAQ Card 1
DAQ Card 2
Windows 7 PC or laptop
Networked Drive
ISA bus backplane crate
This may be too much for this project

Figure 1.6: Concept 2 and its contrast with concept 1.

1.4 Final strategy

The final strategy was as follows:

(i) First analyze the existing code (the old HTBasic code) and figure out all the details. In particular figure out how the experiment data is stored.

(ii) Decide the framework to be used for the Modern GUI.

(iii) Figure out the calibration constants and values as well as methods used, which are required to convert the raw voltages into meaningful data.

(iv) Figure out how the binary data is handled?

(v) After these steps, start with GUI development. Meanwhile, look for interface to communicate the data.

(vi) Decide the interface and change the old code accordingly.

(vii) Testing of the project.

CHAPTER 2 – ANALYSIS OF THE EXISTING CODE -1

The existing control system was written in TRANSERA HTBasic 3.86. The whole code consisted of around 50k lines of sequential code. To find something in this code was like finding a needle in the haystack! So it was very important to decide what I want out of this analysis? The following questions were the prime directive for the analysis:

(i) How a whole test is executed?
(ii) How the data is stored in between the experiment and after the experiment?
(iii) How the raw voltages are converted to the meaningful values?
(iv) Determine the point at which the indentation data first arrives and can be sent through a hardware medium?

The master file for the control system was "XPMASTER"(.bas), so I started with this file. After the analysis I came up with two approaches in which the control system could be described.

2.1 Control Flow

In this approach, I analyzed how the control of the whole system changes in order to perform one indent. I came up with the following result:

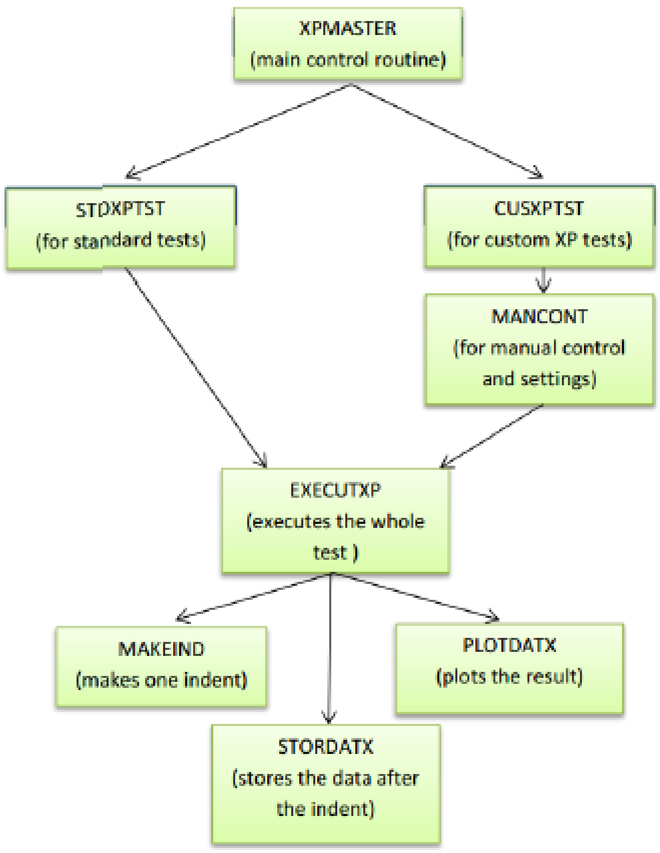


Figure 2.1: Control flow for a single indent

In the above figure, we may see the routine “EXECUTXP” monitors the indentation process, whereas, “MAKEIND” performs one indent.“XPMASTER” controls the whole flow of the program.It provides six options, the two shown actually make indent. For standard XP test STDXPTST is used whereas for custom XP tests CUSXPTST is used. After parameter and shape selection both these routines EXECUTXP to perform indentation. MAKEIND – makes one indent and stores the data in Read(*) and Meas(*) arrays using LICOMSL.STORDATX – stores the following variables in a temp. buffer using GTEXTXP: T$,No,Nv,Vn$(*),Ns,Sn$(*),Sc(*),Logged_data(*), Ctrl_seq(*)

In this way, we got close to answer our first question. Now, the routine “makeind” was analyzed and important variable and methods were taken note of.

2.2 Data flow

To answer the second question it was important to analyze the code from the point of view based only on data. After analysis I came to out with the following result:

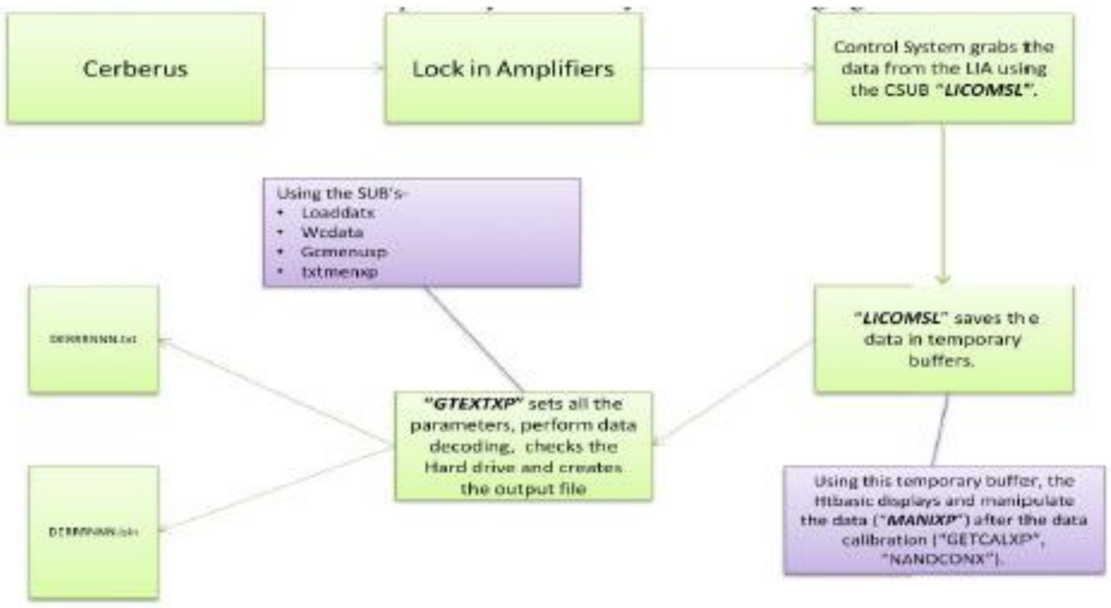
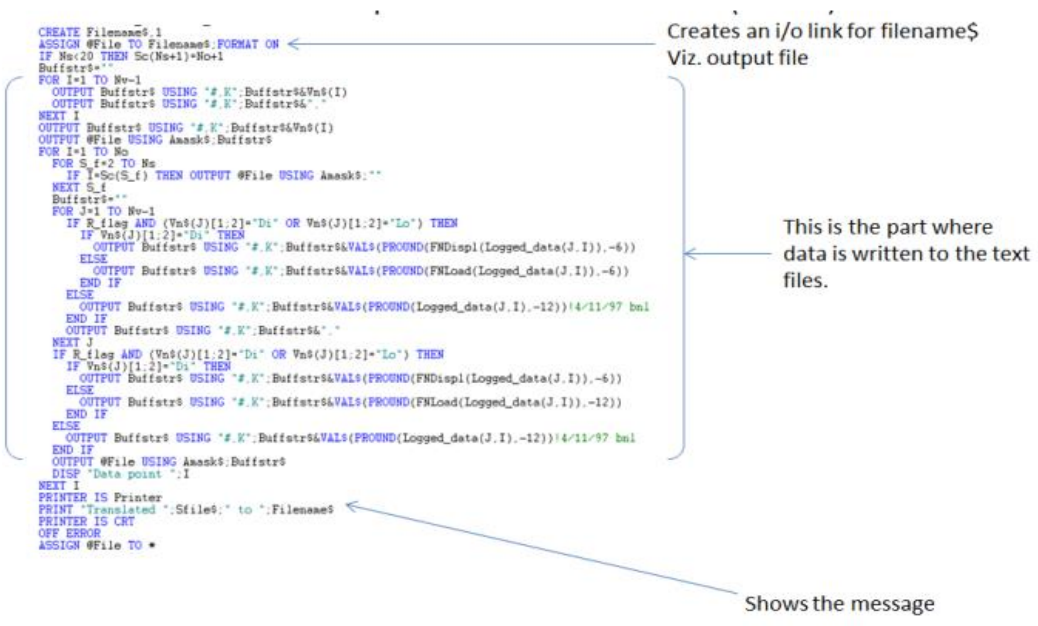

Cerberus
Lock in Amplifiers
Control System grabs the data from the LIA using the CSUB "LICOMSL".
"LICOMSL" saves the data in temporary buffers.
Using this temporary buffer, the Htbasic displays and manipulate the data ("MANIXP") after the data calibration ("GETCALXP", "NANDCONX").
Using the SUB's-
Loaddatx
Wcdata
Gcmenuxp
txtmenxp
"GTEXTXP" sets all the parameters, perform data decoding, checks the Hard drive and creates the output file
DEXRRNNN.txt
DEXRRNNN.bin

Figure 2.2: Data flow for a single indent

It can ce seen in the above figure that the data is until the calibration are done (LICOMSL). After that GTEXTXP(using STORDATX) saves this data to temporary binary files and if the user demands this routine saves the experiment data to the text file. The following figure shows how the data is stored to the text file:



```
CREATE Filename$,1
ASSIGN @File TO Filename$;FORMAT ON
IF Ns<20 THEN Sc(Ns+1)=No+1
Buffstr$=""
FOR I=1 TO Nv-1
  OUTPUT Buffstr$ USING "#,K";Buffstr$&Vn$(I)
  OUTPUT Buffstr$ USING "#,K";Buffstr$&","
NEXT I
OUTPUT Buffstr$ USING "#,K";Buffstr$&Vn$(I)
OUTPUT @File USING Amask$;Buffstr$
FOR I=1 TO No
  FOR S_f=2 TO Ns
    IF I=Sc(S_f) THEN OUTPUT @File USING Amask$;""
  NEXT S_f
  Buffstr$=""
  FOR J=1 TO Nv-1
    IF R_flag AND (Vn$(J)[1;2]="Di" OR Vn$(J)[1;2]="Lo") THEN
      IF Vn$(J)[1;2]="Di" THEN
        OUTPUT Buffstr$ USING "#,K";Buffstr$&VAL$(PROUND(FNDispl(Logged_data(J,I)),-6))
      ELSE
        OUTPUT Buffstr$ USING "#,K";Buffstr$&VAL$(PROUND(FNLoad(Logged_data(J,I)),-6))
      END IF
    ELSE
      OUTPUT Buffstr$ USING "#,K";Buffstr$&VAL$(PROUND(Logged_data(J,I),-12))!4/11/97 bnl
    END IF
    OUTPUT Buffstr$ USING "#,K";Buffstr$&","
  NEXT J
  IF R_flag AND (Vn$(J)[1;2]="Di" OR Vn$(J)[1;2]="Lo") THEN
    IF Vn$(J)[1;2]="Di" THEN
      OUTPUT Buffstr$ USING "#,K";Buffstr$&VAL$(PROUND(FNDispl(Logged_data(J,I)),-6))
    ELSE
      OUTPUT Buffstr$ USING "#,K";Buffstr$&VAL$(PROUND(FNLoad(Logged_data(J,I)),-12))
    END IF
  ELSE
    OUTPUT Buffstr$ USING "#,K";Buffstr$&VAL$(PROUND(Logged_data(J,I),-12))!4/11/97 bnl
  END IF
  OUTPUT @File USING Amask$;Buffstr$
  DISP "Data point ";I
NEXT I
PRINTER IS Printer
PRINT "Translated ";Sfile$;" to ";Filename$
PRINTER IS CRT
OFF ERROR
ASSIGN @File TO *
```

Creates an i/o link for filename$
Viz. output file
This is the part where data is written to the text files.
Shows the message

Figure 2.3: Portion of “GTEXTXP” where data is actually written to the binary file.

In this way, we were able to answer the second question. Now in the next chapter discusses about the selection of framework and we will get back to out four question in the subsequent chapters.

CHAPTER 3 – DECIDING THE FRAMEWORK

In this chapter, we compare the environments available for scientific and numerical computations.

The most popular environments for this purpose is Matlab, but due to project nature it was necessary to use an open source environment. The best open source alternative are :-

1. GNU Octave
2. FreeMat
3. Scilab
4. Pythonxy

The pros and cons of each environment were taken and then the final environment was decided after discussion with my supervisors. The comparision was made on the basis of following points:

1. Ease of installation – ideally available for Win98/XP/Vista/Win7 using self-extracting installers.
2. Well integrated with strong plugin support.
3. Must be strong in computations.
4. Easy to compile / run.
5. User Interface.
6. I/O – file, DAQ and GPIB support if possible.
7. Documentation – well-written manuals and help files are essential.
8. Support – good host site with evidence that there are plenty of other users.

3.1 COMPARISION BETWEEN GNU OCTAVE, FREEMAT, SCILAB AND PYTHONXY

1. GNU Octave:

   GNU Octave is a high-level interpreted language, primarily intended for numerical computations. It provides capabilities for the numerical solution of linear and nonlinear problems, and for performing other numerical experiments. It also provides extensive graphics capabilities for data visualization and manipulation. Octave is normally used through its interactive command line interface, but it can also be used to write non-interactive programs. The Octave language is quite similar to Matlab so that most programs are easily portable.

2. FreeMat:

   FreeMat is a free environment for rapid engineering and scientific prototyping and data processing. It is similar to MATLAB from Mathworks, and IDL from Research Systems, but is Open Source. FreeMat is available under the GPL license.

3. Scilab:

Scilab is free and open source software for numerical computation providing a powerful computing environment for engineering and scientific applications. Scilab is released as open source under the CeCILL license (GPL compatible), and is available for download free of charge. Scilab is available under GNU/Linux, Mac OS X and Windows XP/Vista/7/8.

Scilab includes hundreds of mathematical functions. It has a high level programming language allowing access to advanced data structures, 2-D and 3-D graphical functions.

4. Pythonxy:

Python(x,y) is a free scientific and engineering development software for numerical computations, data analysis and data visualization based on Python programming language, Qt graphical user interfaces and Spyder interactive scientific development environment.

After heavy discussion with my supervisors and analysis of information available on the internet, I have graded each environment on 8 factors on a scale of 10.

Environment	Installation	Plugins	Numerical computations	Ease to compile	UI	I/O	Document ation	Online suppor
Matlab	7	10	10	10	9	9	10	10
Octave	8	6	9	8	7	8	9	10
FreeMat	9	5	8	7	6	8	7	8
Pythonxy	8	10	9	10	9	9	10	10
Scilab	9	7	9	10	9	8	8	10

Table 3.1 : comaprision between different available options

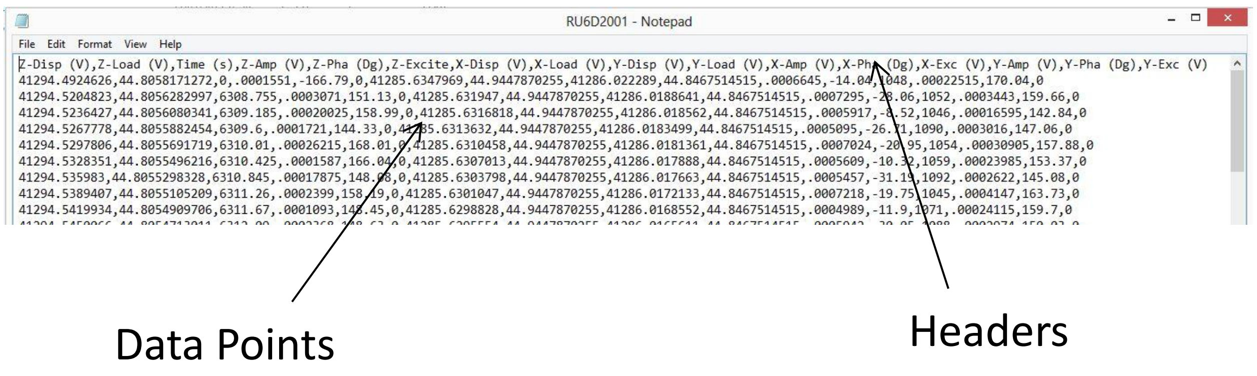
After deep evaluation of the available options due to its slight advantage over Matlab and increasing use in the scientific community, Pythonxy is selected as the final environment.

CHAPTER 4 – ANALYSIS OF THE EXISTING CODE - 2

In this chapter, we will analyze the contents of the text file and discuss the calibration constants and values which came out after deeply analzing the code.

4.1 The “TEXT” file

The text file denotes a file containing the experiment data (not calibrated) in ASCII format. The control system produces this kind of file when the user demands for it. In normal case, data is stored only in a binary file.



```
Z-Disp (V),Z-Load (V),Time (s),Z-Amp (V),Z-Pha (Dg),Z-Excite,X-Disp (V),X-Load (V),Y-Disp (V),Y-Load (V),X-Amp (V),X-Pha (Dg),X-Exc (V),Y-Amp (V),Y-Pha (Dg),Y-Exc (V)
41294.4924626,44.8058171272,0,.0001551,-166.79,0,41285.6347969,44.9447870255,41286.022289,44.8467514515,.0006645,-14.04,1048,.00022515,170.04,0
41294.5204823,44.8056282997,6308.755,.0003071,151.13,0,41285.631947,44.9447870255,41286.0188641,44.8467514515,.0007295,-28.06,1052,.0003443,159.66,0
41294.5236427,44.8056080341,6309.185,.00020025,158.99,0,41285.6316818,44.9447870255,41286.018562,44.8467514515,.0005917,-8.52,1046,.00016595,142.84,0
41294.5267778,44.8055882454,6309.6,.0001721,144.33,0,41285.6313632,44.9447870255,41286.0183499,44.8467514515,.0005095,-26.71,1090,.0003016,147.06,0
41294.5297806,44.8055691719,6310.01,.00026215,168.01,0,41285.6310458,44.9447870255,41286.0181361,44.8467514515,.0007024,-20.95,1054,.00030905,157.88,0
41294.5328351,44.8055496216,6310.425,.0001587,166.04,0,41285.6307013,44.9447870255,41286.017888,44.8467514515,.0005609,-10.32,1059,.00023985,153.37,0
41294.535983,44.8055298328,6310.845,.00017875,148.08,0,41285.6303798,44.9447870255,41286.017663,44.8467514515,.0005457,-31.19,1092,.0002622,145.08,0
41294.5389407,44.8055105209,6311.26,.0002399,158.19,0,41285.6301047,44.9447870255,41286.0172133,44.8467514515,.0007218,-19.75,1045,.0004147,163.73,0
41294.5419934,44.8054909706,6311.67,.0001093,148.45,0,41285.6298828,44.9447870255,41286.0168552,44.8467514515,.0004989,-11.9,1071,.00024115,159.7,0
```

Figure 4.1 : The text file produced by the ceberus control system.

The text file contains two major portions: Header – names of all the channels or variables and
Data points – data for these channels. But this data can not be used to produce the final results. It uses lots of calibration settings to produce the final results.

4.2 Calibration Constants and Values

In this section we will deal with the issue of getting the final results from the data in the text file. The data coming from the system is raw voltage value.The Cerberus system uses various constants to convert the raw voltages into meaningful data. Several calibration constants are involved. These are constant values that does not depend on the data and referred as Calibration Constant, henceforth. But some of the calibration do depend on the data , these will be referred as calibration values. See **Appendix I and II** for these values. These values were figured out after deep analysis of code and referencing standard nanoindentaion sources.

4.3 Data Calculation

Now , we have raw data, calibration constant and values, the final job is to figure out how final results are calculated using these values. Again it required a deep analysis of the code and the result is shown in the following table. The first column denote the quantity to be calculated and second column shows how this quantity will be calculated.

Quantity	Description
Time (s)[i]	Data.time[i] – data.time[2] i from 2 to max
Z-Disp R3 (V)[i]	Data.ZDisp[i] – 5 –Z2*10 – (Z1+1)*10000
Z-Disp R3 Zeroed (V)[i]	Z-Disp R3 [i] - Z-Disp R3 [1]
Z-Disp Total (nm)[i]	Z-Disp R3 Zeroed[i] * Z3 / 2^(Z1)
X-Disp R3 (V)[i]	Data.XDisp[i] – 5 – X2 *10 – (X1+1)*10000
X-Disp R3 Zeroed (V)[i]	X-Disp R3[i] - X-Disp R3[1]
X-Disp(nm)[i]	X-Disp R3 Zeroed[i]*X3/2^(X1)
Y-Disp R3 (V)[i]	Data.YDisp[i] – 5 –Y2*10 – (Y1+1)*10000
Y-Disp R3 Zeroed (V)[i]	Y-Disp R3[i] - Y-Disp R3[1]
Y-Disp(nm)[i]	Y-Disp R3 Zeroed[i]*Y3/2^(Y1)
Z-Disp Zeroed(nm)[i]	Z-Disp Total[i] – zero_disp_z
x-Disp Zeroed (nm) [i]	x-Disp Total[i] – zero_disp_x
y-Disp Zeroed (nm) [i]	y-Disp Total[i] – zero_disp_y
Time Zeroed (s) [i]	Time[i] – zero_time
Z-Amplitude[i]	Data.ZAmp[i]*Z4/2^(Z1)
Z-Force (μN)[i]	[1] -> data.ZExcite[1]*C23*C26 [i] -> data.ZExcite[i]*Z8*Z7
Z-Load (V)[i]	Data.Zload[i]- (Z5+1)*10 -5
Z-Load Zeroed (V)[i]	-(ZLoad[i]-zero_load)
Z-Load (mN)[i]	Z-Load Zeroed [i]*Z6/1000
Z-Load Corrected for Kls (mN)[i]	Z-Load (mN)[i] – SLOPE(load_for_kls, disp_for_kls)* Z-Disp Zeroed (nm)[i]
Raw Z Phase (Deg)[i]	Data.ZPha[i]*1
Corrected Z Phase (Deg)[i]	Raw Z Phase (Deg)[i]-(-0.0844*125-0.0596)
COS Z Phase (_)[i]	COS(Corrected Z Phase (Deg)[i]*pi/180)
F/X CosPHI (N/m) [i]	Z-Force (μN)[i]/ Z-Amplitude[i]*COS(Corrected Z Phase (Deg)[i]*PI()/180)*1000
F/X CosPHI-{k-mw^2} (N/m) [i]	F/X CosPHI (N/m) [i]-C7Z-C27Z*2*PI()*125*2*PI()*125
Z-Stiffness	1/(1/ F/X CosPHI-{k-mw^2} (N/m) [i]-1/C33Z)

(N/m)[i]	
Z-P/S^2 (m^2/N)[i]	Z-Load Corrected for Kls (mN)[i]/ Z-Stiffness (N/m)[i]/ Z-Stiffness (N/m)[i]/1000
X-Amplitude (nm)[i]	Data. X-Amp (V)[i]*X4/2^X1
Y-Amplitude (nm)[i]	Data. X-Amp (V)[i]*Y4/2^Y1
X-Force (μN)[i]	XExc(v)[i]*C23X*C26X
Y-Force (μN)[i]	YExc(v)[i]*C23Y*C26Y
Z-F/X (N/m)[i]	Z-Force (μN)[i]/ Z-Amplitude[i]*1000
X-F/X (N/m)[i]	X-Force (μN)[i]/ X-Amplitude (nm)[i]*1000
Y-F/X (N/m)[i]	Y-Force (μN)[i]/Y-Amplitude (nm)[i]*1000
X-Harmonic F for Const X/Quasi-static Z-Force (_)[i]	X-Force (μN)[i]/ Z-Load Corrected for Kls (mN)[i]/1000
Y-Harmonic F for Const X/Quasi-static Z-Force (_)[i]	Y-Force (μN)[i]/ Z-Load Corrected for Kls (mN)[i]/1000
Raw X Phase (Deg)[i]	Data. X-Pha (Dg)[i]*(1)
Corrected X Phase (Deg)[i]	Raw X Phase (Deg)[i]-(-0.0844*125-0.0596)
COS X Phase (_)[i]	COS(Corrected X Phase (Deg)[i]*PI()/180)
Raw Y Phase (Deg)[i]	Data. Y-Pha (Dg)[i]*(1)
Corrected Y Phase (Deg)[i]	Raw Y Phase (Deg)[i]-(-0.0844*125-0.0596)
COS Y Phase (_)[i]	COS(Corrected Y Phase (Deg)[i]*PI()/180)
X-F/X CosPHI (N/m)[i]	X-F/X (N/m)[i]* COS X Phase (_)[i]
X-F/X CosPHI-{k-mw^2} (N/m)[i]	X-F/X CosPHI (N/m)[i]-(C7X-C27X*2*PI()*125*2*PI()*125)
X Stiffness (N/m)[i]	1/(1/ X-F/X CosPHI-{k-mw^2} (N/m)[i]-1/C33X)
Y-F/x CosPHI (N/m)[i]	Y-F/X (N/m)[i]* COS Y Phase (_)[i]
Y-F/x CosPHI-{k-mw^2} (N/m)[i]	Y-F/x CosPHI (N/m)[i]-(C7Y-C27Y*2*PI()*125*2*PI()*125)

Y Stiffness (N/m)[i]	1/(1/ Y-F/x CosPHI-{k-mw^2} (N/m)[i]-1/C33Y)
X P/S^2 (m^2/N)[i]	Z-Load Corrected for Kls (mN)[i]/ X Stiffness (N/m)[i]/ X Stiffness (N/m)[i]/1000
Y P/S^2 (m^2/N) [i]	Z-Load Corrected for Kls (mN)[i]/ Y Stiffness (N/m)[i]/ Y Stiffness (N/m)[i]/1000
X-SinPHI (_)[i]	SIN(Corrected X Phase (Deg)[i]*PI()/180)
X-F/X SinPHI (N/m)[i]	X-F/X (N/m)[i]* X-SinPHI (_)[i]
ciw (N/m)[i]	
Cw/S X-Axis (_)[i]	X-F/X SinPHI-ciw (N/m)[i]/ X Stiffness (N/m)[i]
Y-SinPHI (_)[i]	SIN(Corrected Y Phase (Deg)[i]*PI()/180)
Y-F/X SinPHI (N/m)[i]	Y-F/X (N/m)[i]* Y-SinPHI (_)[i]
Y-F/X SinPHI-ciw (N/m)[i]	Y-F/X SinPHI (N/m)[i]-C28Y*125*2*PI()
Cw/S Y-Axis (_)[i]	Y-F/X SinPHI-ciw (N/m)[i]/ Y Stiffness (N/m)[i]

Table 4.1 : Final results and their formulas

Now, we have everything we need in order to generate the final results from the text file. Using the Calibration constants and values discussed earlier along with these formulas, we can calculate the final results and get the desired plots. In order to test the observed values, I decided to make a small program which generates the load-displacement plot from the given text file. The following figure shows the output of the program:

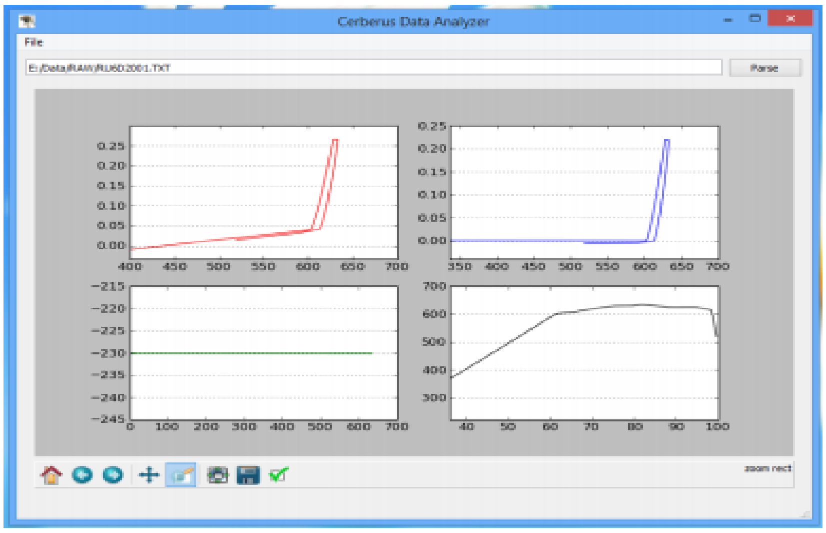


Figure 4.2 : Output of the code parsing the text file.

I used python(x,y) as the framework. For GUI, pyqt4 has been used. For graph plotting MatPlotLib was used. For numerical computations, NumPy was used.

CHAPTER 5 – DECODING THE BINARY DATA

Until this point I have developed the code to read the data from the text file and produce the final results. Now, in order to enable the python code to read the data live from the old control system, via some interface discussed later, we must know how to process the binary files. It is so because the data is stored in the binary format in a temporary buffer during the experiment.

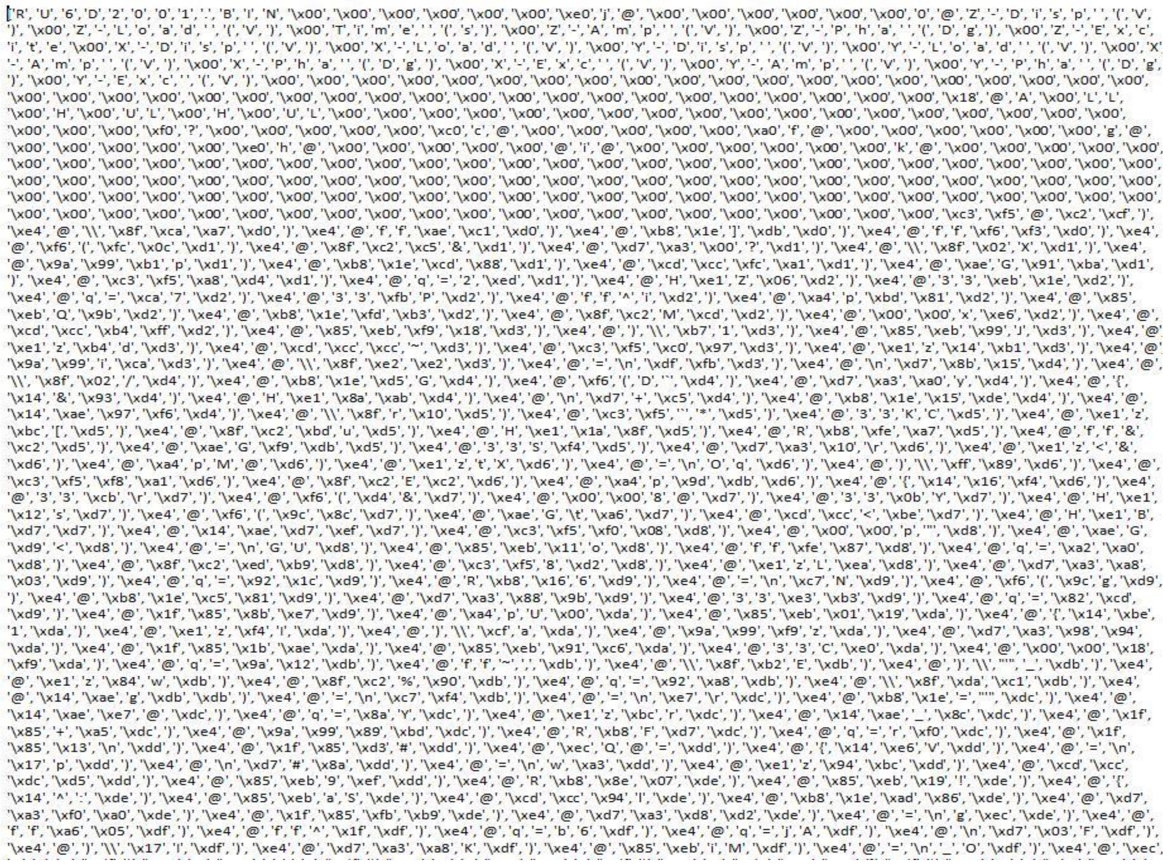
5.1 BINARY FILE

5.1.1 Contents of the binary file.

In order to get the binary data, I wrote a python script. After observing the existing code, it became clear that data was written in a sequential manner.

1. The file name (RU6D2001.bin , in this case).
2. Channels in the data stream.
3. The actual readings.(logged_data(*))
4. Command sequence.

Python interpreter has methods to directly access the binary file and using those methods file is explored. The following figure shows the actual content of the binary file:



```
['R', 'U', '6', 'D', '2', '0', '0', '1', '.', 'B', 'I', 'N', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\xe0', 'j', '@', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '0', '@', 'Z', '-', 'D', 'i', 's', 'p', ' ', '(', 'V',
')', '\x00', 'Z', '-', 'L', 'o', 'a', 'd', ' ', '(', 'V', ')', '\x00', 'T', 'i', 'm', 'e', ' ', '(', 's', ')', '\x00', 'Z', '-', 'A', 'm', 'p', ' ', '(', 'V', ')', '\x00', 'Z', '-', 'P', 'h', 'a', ' ', '(', 'D', 'g', ')', '\x00', 'Z', '-', 'E', 'x', 'c',
'i', 't', 'e', '\x00', 'X', '-', 'D', 'i', 's', 'p', ' ', '(', 'V', ')', '\x00', 'X', '-', 'L', 'o', 'a', 'd', ' ', '(', 'V', ')', '\x00', 'Y', '-', 'D', 'i', 's', 'p', ' ', '(', 'V', ')', '\x00', 'Y', '-', 'L', 'o', 'a', 'd', ' ', '(', 'V', ')', '\x00', 'X'
'-', 'A', 'm', 'p', ' ', '(', 'V', ')', '\x00', 'X', '-', 'P', 'h', 'a', ' ', '(', 'D', 'g', ')', '\x00', 'X', '-', 'E', 'x', 'c', ' ', '(', 'V', ')', '\x00', 'Y', '-', 'A', 'm', 'p', ' ', '(', 'V', ')', '\x00', 'Y', '-', 'P', 'h', 'a', ' ', '(', 'D', 'g',
')', '\x00', 'Y', '-', 'E', 'x', 'c', ' ', '(', 'V', ')', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
'\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x18', '@', 'A', '\x00', 'L', 'L',
'\x00', 'H', '\x00', 'U', 'L', '\x00', 'H', '\x00', 'U', 'L', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
'\x00', '\x00', '\x00', '\xf0', '?', '\x00', '\x00', '\x00', '\x00', '\x00', '\xc0', 'c', '@', '\x00', '\x00', '\x00', '\x00', '\x00', '\xa0', 'f', '@', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', 'g', '@',
'\x00', '\x00', '\x00', '\x00', '\x00', '\xe0', 'h', '@', '\x00', '\x00', '\x00', '\x00', '\x00', '@', 'i', '@', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', 'k', '@', '\x00', '\x00', '\x00', '\x00', '\x00',
'\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
'\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
'\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
'\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\x00', '\xc3', '\xf5', '@', '\xc2', '\xcf', ')',
'\xe4', '@', '\\', '\x8f', '\xca', '\xa7', '\xd0', ')', '\xe4', '@', 'f', 'f', '\xae', '\xc1', '\xd0', ')', '\xe4', '@', '\xb8', '\x1e', ']', '\xdb', '\xd0', ')', '\xe4', '@', 'f', 'f', '\xf6', '\xf3', '\xd0', ')', '\xe4',
'@', '\xf6', '(', '\xfc', '\x0c', '\xd1', ')', '\xe4', '@', '\x8f', '\xc2', '\xc5', '&', '\xd1', ')', '\xe4', '@', '\xd7', '\xa3', '\x00', '?', '\xd1', ')', '\xe4', '@', '\\', '\x8f', '\x02', 'X', '\xd1', ')', '\xe4',
'@', '\x9a', '\x99', '\xb1', 'p', '\xd1', ')', '\xe4', '@', '\xb8', '\x1e', '\xcd', '\x88', '\xd1', ')', '\xe4', '@', '\xcd', '\xcc', '\xfc', '\xa1', '\xd1', ')', '\xe4', '@', '\xae', 'G', '\x91', '\xba', '\xd1',
')', '\xe4', '@', '\xc3', '\xf5', '\xa8', '\xd4', '\xd1', ')', '\xe4', '@', 'q', '=', '2', '\xed', '\xd1', ')', '\xe4', '@', 'H', '\xe1', 'Z', '\x06', '\xd2', ')', '\xe4', '@', '3', '3', '\xeb', '\x1e', '\xd2', ')',
'\xe4', '@', 'q', '=', '\xca', '7', '\xd2', ')', '\xe4', '@', '3', '3', '\xfb', 'P', '\xd2', ')', '\xe4', '@', 'f', 'f', '^', 'i', '\xd2', ')', '\xe4', '@', '\xa4', 'p', '\xbd', '\x81', '\xd2', ')', '\xe4', '@', '\x85',
'\xeb', 'Q', '\x9b', '\xd2', ')', '\xe4', '@', '\xb8', '\x1e', '\xfd', '\xb3', '\xd2', ')', '\xe4', '@', '\x8f', '\xc2', 'M', '\xcd', '\xd2', ')', '\xe4', '@', '\x00', '\x00', 'x', '\xe6', '\xd2', ')', '\xe4', '@',
'\xcd', '\xcc', '\xb4', '\xff', '\xd2', ')', '\xe4', '@', '\x85', '\xeb', '\xf9', '\x18', '\xd3', ')', '\xe4', '@', ')', '\\', '\xb7', '1', '\xd3', ')', '\xe4', '@', '\x85', '\xeb', '\x99', 'J', '\xd3', ')', '\xe4', '@'
'\xe1', 'z', '\xb4', 'd', '\xd3', ')', '\xe4', '@', '\xcd', '\xcc', '\xcc', '~', '\xd3', ')', '\xe4', '@', '\xc3', '\xf5', '\xc0', '\x97', '\xd3', ')', '\xe4', '@', '\xe1', 'z', '\x14', '\xb1', '\xd3', ')', '\xe4', '@'
'\x9a', '\x99', 'i', '\xca', '\xd3', ')', '\xe4', '@', '\\', '\x8f', '\xe2', '\xe2', '\xd3', ')', '\xe4', '@', '=', '\n', '\xdf', '\xfb', '\xd3', ')', '\xe4', '@', '\n', '\xd7', '\x8b', '\x15', '\xd4', ')', '\xe4', '@',
'\\', '\x8f', '\x02', '/', '\xd4', ')', '\xe4', '@', '\xb8', '\x1e', '\xd5', 'G', '\xd4', ')', '\xe4', '@', '\xf6', '(', 'D', '`', '\xd4', ')', '\xe4', '@', '\xd7', '\xa3', '\xa0', 'y', '\xd4', ')', '\xe4', '@', '{',
'\x14', '&', '\x93', '\xd4', ')', '\xe4', '@', 'H', '\xe1', '\x8a', '\xab', '\xd4', ')', '\xe4', '@', '\n', '\xd7', '+', '\xc5', '\xd4', ')', '\xe4', '@', '\xb8', '\x1e', '\x15', '\xde', '\xd4', ')', '\xe4', '@',
'\x14', '\xae', '\x97', '\xf6', '\xd4', ')', '\xe4', '@', '\\', '\x8f', 'r', '\x10', '\xd5', ')', '\xe4', '@', '\xc3', '\xf5', '`', '*', '\xd5', ')', '\xe4', '@', '3', '3', 'K', 'C', '\xd5', ')', '\xe4', '@', '\xe1', 'z',
'\xbc', '[', '\xd5', ')', '\xe4', '@', '\x8f', '\xc2', '\xbd', 'u', '\xd5', ')', '\xe4', '@', 'H', '\xe1', '\x1a', '\x8f', '\xd5', ')', '\xe4', '@', 'R', '\xb8', '\xfe', '\xa7', '\xd5', ')', '\xe4', '@', 'f', 'f', '&',
'\xc2', '\xd5', ')', '\xe4', '@', '\xae', 'G', '\xf9', '\xdb', '\xd5', ')', '\xe4', '@', '3', '3', 'S', '\xf4', '\xd5', ')', '\xe4', '@', '\xd7', '\xa3', '\x10', '\r', '\xd6', ')', '\xe4', '@', '\xe1', 'z', '<', '&',
'\xd6', ')', '\xe4', '@', '\xa4', 'p', 'M', '@', '\xd6', ')', '\xe4', '@', '\xe1', 'z', 't', 'X', '\xd6', ')', '\xe4', '@', '=', '\n', 'O', 'q', '\xd6', ')', '\xe4', '@', ')', '\\', '\xff', '\x89', '\xd6', ')', '\xe4', '@',
'\xc3', '\xf5', '\xf8', '\xa1', '\xd6', ')', '\xe4', '@', '\x8f', '\xc2', 'E', '\xc2', '\xd6', ')', '\xe4', '@', '\xa4', 'p', '\x9d', '\xdb', '\xd6', ')', '\xe4', '@', '{', '\x14', '\x16', '\xf4', '\xd6', ')', '\xe4',
'@', '3', '3', '\xcb', '\r', '\xd7', ')', '\xe4', '@', '\xf6', '(', '\xd4', '&', '\xd7', ')', '\xe4', '@', '\x00', '\x00', '8', '@', '\xd7', ')', '\xe4', '@', '3', '3', '\x0b', 'Y', '\xd7', ')', '\xe4', '@', 'H', '\xe1',
'\x12', 's', '\xd7', ')', '\xe4', '@', '\xf6', '(', '\x9c', '\x8c', '\xd7', ')', '\xe4', '@', '\xae', 'G', '\t', '\xa6', '\xd7', ')', '\xe4', '@', '\xcd', '\xcc', '<', '\xbe', '\xd7', ')', '\xe4', '@', 'H', '\xe1', 'B',
'\xd7', '\xd7', ')', '\xe4', '@', '\x14', '\xae', '\xd7', '\xef', '\xd7', ')', '\xe4', '@', '\xc3', '\xf5', '\xf0', '\x08', '\xd8', ')', '\xe4', '@', '\x00', '\x00', 'p', '"', '\xd8', ')', '\xe4', '@', '\xae', 'G',
'\xd9', '<', '\xd8', ')', '\xe4', '@', '=', '\n', 'G', 'U', '\xd8', ')', '\xe4', '@', '\x85', '\xeb', '\x11', 'o', '\xd8', ')', '\xe4', '@', 'f', 'f', '\xfe', '\x87', '\xd8', ')', '\xe4', '@', 'q', '=', '\xa2', '\xa0',
'\xd8', ')', '\xe4', '@', '\x8f', '\xc2', '\xed', '\xb9', '\xd8', ')', '\xe4', '@', '\xc3', '\xf5', '8', '\xd2', '\xd8', ')', '\xe4', '@', '\xe1', 'z', 'L', '\xea', '\xd8', ')', '\xe4', '@', '\xd7', '\xa3', '\xa8',
'\x03', '\xd9', ')', '\xe4', '@', 'q', '=', '\x92', '\x1c', '\xd9', ')', '\xe4', '@', 'R', '\xb8', '\x16', '6', '\xd9', ')', '\xe4', '@', '=', '\n', '\xc7', 'N', '\xd9', ')', '\xe4', '@', '\xf6', '(', '\x9c', 'g', '\xd9',
')', '\xe4', '@', '\xb8', '\x1e', '\xc5', '\x81', '\xd9', ')', '\xe4', '@', '\xd7', '\xa3', '\x88', '\x9b', '\xd9', ')', '\xe4', '@', '3', '3', '\xe3', '\xb3', '\xd9', ')', '\xe4', '@', 'q', '=', '\x82', '\xcd',
'\xd9', ')', '\xe4', '@', '\x1f', '\x85', '\x8b', '\xe7', '\xd9', ')', '\xe4', '@', '\xa4', 'p', 'U', '\x00', '\xda', ')', '\xe4', '@', '\x85', '\xeb', '\x01', '\x19', '\xda', ')', '\xe4', '@', '{', '\x14', '\xbe',
'1', '\xda', ')', '\xe4', '@', '\xe1', 'z', '\xf4', 'I', '\xda', ')', '\xe4', '@', ')', '\\', '\xcf', 'a', '\xda', ')', '\xe4', '@', '\x9a', '\x99', '\xf9', 'z', '\xda', ')', '\xe4', '@', '\xd7', '\xa3', '\x98', '\x94',
'\xda', ')', '\xe4', '@', '\x1f', '\x85', '\x1b', '\xae', '\xda', ')', '\xe4', '@', '\x85', '\xeb', '\x91', '\xc6', '\xda', ')', '\xe4', '@', '3', '3', 'C', '\xe0', '\xda', ')', '\xe4', '@', '\x00', '\x00', '\x18',
'\xf9', '\xda', ')', '\xe4', '@', 'q', '=', '\x9a', '\x12', '\xdb', ')', '\xe4', '@', 'f', 'f', '~', ',', '\xdb', ')', '\xe4', '@', '\\', '\x8f', '\xb2', 'E', '\xdb', ')', '\xe4', '@', ')', '\\', '"', '_', '\xdb', ')', '\xe4',
'@', '\xe1', 'z', '\x84', 'w', '\xdb', ')', '\xe4', '@', '\x8f', '\xc2', '%', '\x90', '\xdb', ')', '\xe4', '@', 'q', '=', '\x92', '\xa8', '\xdb', ')', '\xe4', '@', '\\', '\x8f', '\xda', '\xc1', '\xdb', ')', '\xe4',
'@', '\x14', '\xae', 'g', '\xdb', '\xdb', ')', '\xe4', '@', '=', '\n', '\xc7', '\xf4', '\xdb', ')', '\xe4', '@', '=', '\n', '\xe7', '\r', '\xdc', ')', '\xe4', '@', '\xb8', '\x1e', '=', '"', '\xdc', ')', '\xe4', '@',
'\x14', '\xae', '\xe7', '@', '\xdc', ')', '\xe4', '@', 'q', '=', '\x8a', 'Y', '\xdc', ')', '\xe4', '@', '\xe1', 'z', '\xbc', 'r', '\xdc', ')', '\xe4', '@', '\x14', '\xae', '_', '\x8c', '\xdc', ')', '\xe4', '@', '\x1f',
'\x85', '+', '\xa5', '\xdc', ')', '\xe4', '@', '\x9a', '\x99', '\x89', '\xbd', '\xdc', ')', '\xe4', '@', 'R', '\xb8', 'F', '\xd7', '\xdc', ')', '\xe4', '@', 'q', '=', 'r', '\xf0', '\xdc', ')', '\xe4', '@', '\x1f',
'\x85', '\x13', '\n', '\xdd', ')', '\xe4', '@', '\x1f', '\x85', '\xd3', '#', '\xdd', ')', '\xe4', '@', '\xec', 'Q', '@', '=', '\xdd', ')', '\xe4', '@', '{', '\x14', '\xe6', 'V', '\xdd', ')', '\xe4', '@', '=', '\n',
'\x17', 'p', '\xdd', ')', '\xe4', '@', '\n', '\xd7', '#', '\x8a', '\xdd', ')', '\xe4', '@', '=', '\n', 'w', '\xa3', '\xdd', ')', '\xe4', '@', '\xe1', 'z', '\x94', '\xbc', '\xdd', ')', '\xe4', '@', '\xcd', '\xcc',
'\xdc', '\xd5', '\xdd', ')', '\xe4', '@', '\x85', '\xeb', '9', '\xef', '\xdd', ')', '\xe4', '@', 'R', '\xb8', '\x8e', '\x07', '\xde', ')', '\xe4', '@', '\x85', '\xeb', '\x19', '!', '\xde', ')', '\xe4', '@', '{',
'\x14', '^', ':', '\xde', ')', '\xe4', '@', '\x85', '\xeb', 'a', 'S', '\xde', ')', '\xe4', '@', '\xcd', '\xcc', '\x94', 'l', '\xde', ')', '\xe4', '@', '\xb8', '\x1e', '\xad', '\x86', '\xde', ')', '\xe4', '@', '\xd7',
'\xa3', '\xf0', '\xa0', '\xde', ')', '\xe4', '@', '\x1f', '\x85', '\xfb', '\xb9', '\xde', ')', '\xe4', '@', '\xd7', '\xa3', '\xd8', '\xd2', '\xde', ')', '\xe4', '@', '=', '\n', 'g', '\xec', '\xde', ')', '\xe4', '@',
'f', 'f', '\xa6', '\x05', '\xdf', ')', '\xe4', '@', 'f', 'f', '^', '\x1f', '\xdf', ')', '\xe4', '@', 'q', '=', 'b', '6', '\xdf', ')', '\xe4', '@', 'q', '=', 'j', 'A', '\xdf', ')', '\xe4', '@', '\n', '\xd7', '\x03', 'F', '\xdf', ')',
'\xe4', '@', ')', '\\', '\x17', 'I', '\xdf', ')', '\xe4', '@', '\xd7', '\xa3', '\xa8', 'K', '\xdf', ')', '\xe4', '@', '\x85', '\xeb', 'i', 'M', '\xdf', ')', '\xe4', '@', '=', '\n', '_', 'O', '\xdf', ')', '\xe4', '@', '\xec',
```

Figure 5.1 : Contents of the binary file.

What is written in the binary file? The binary file is written in 3 steps:

1. Filename, number of readings, number of channels, name of the channels….
2. Logged_data(*) contains all the readings.
3. Ctrl_seq$(*) control sequence of the experiment. HTBasic uses a different kind of protocol for file writing:

- It writes the string variable as it is.
- It writes the numerals in **standard numeric representation** format.
- It writes the arrays in the row major order.

5.2 Getting data from binary file

5.2.1 Approaches

We had two ways to get the data from binary file:

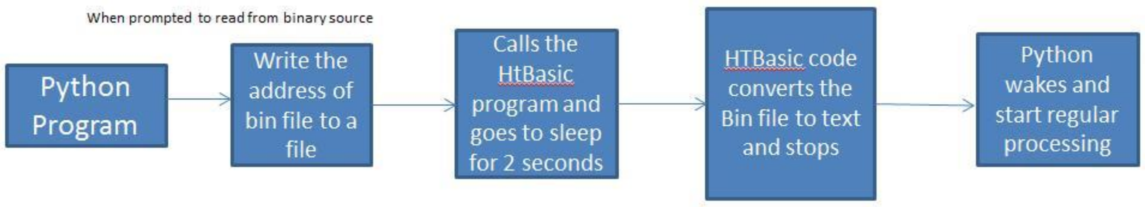
5.3 To write a python script to read the file and the standard representation, as python does not have a module to read these files directly, but NumPy module has a function NumPyfromstring('string') to read the number one at a time.

2. The second way is to write a HTBasic routine to read the files and call that routine from the python program.

Since the total number of reading is around 3500, so it's not efficient to convert every reading from the standard numeric format to a float. The second way is better considering the time and space complexity.

5.3 Final algorithm

The final algorithm is depicted in the following figure:



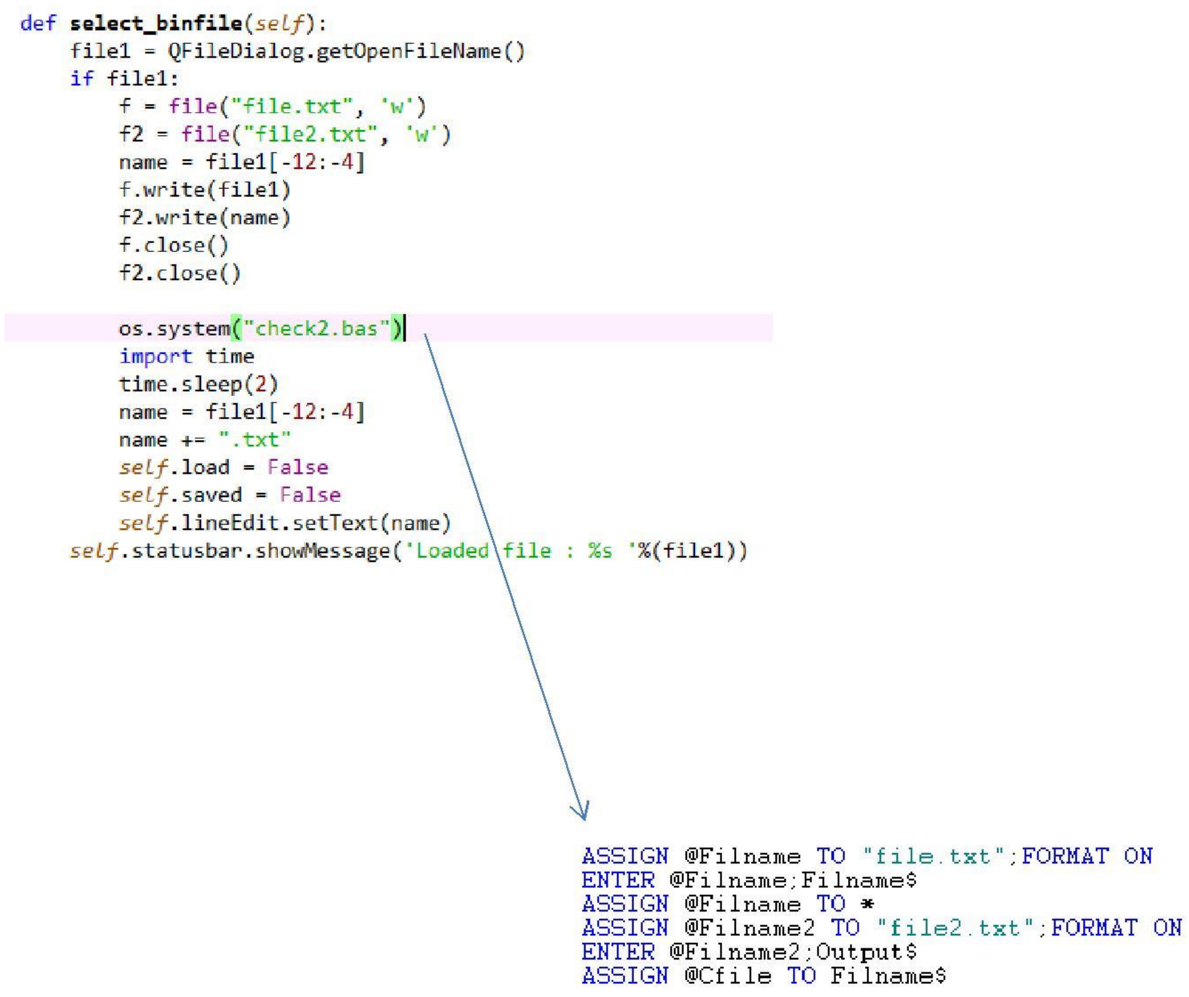
When prompted to read from binary source
Python Program
Write the address of bin file to a file
Calls the HtBasic program and goes to sleep for 2 seconds
HTBasic code converts the Bin file to text and stops
Python wakes and start regular processing

Figure 5.2 : Approach to read binary file

When to python program, Cerberus Data Analyser(CDA), is prompted to receive binary data from the disk, it reads the address of the binary file and writes it to a file. After writing the address the python program calls a HTBasic routine and goes to sleep for 2 seconds. The HTBasic code reads the address of the bin file from the file and converts the binary file to a text file as produced by the original control system. Then CDA wakes up and process this text file and produce the result.

5.4 Interplay between Python and HTBasic

The following figure shows the interplay between CDA and HTBasic codes. On the left side is the python code and right one on the HTBasic code.



```
def select_binfile(self):
    file1 = QFileDialog.getOpenFileName()
    if file1:
        f = file("file.txt", 'w')
        f2 = file("file2.txt", 'w')
        name = file1[-12:-4]
        f.write(file1)
        f2.write(name)
        f.close()
        f2.close()

        os.system("check2.bas")
        import time
        time.sleep(2)
        name = file1[-12:-4]
        name += ".txt"
        self.load = False
        self.saved = False
        self.lineEdit.setText(name)
    self.statusbar.showMessage('Loaded file : %s '%(file1))
```

```
ASSIGN @Filname TO "file.txt";FORMAT ON
ENTER @Filname;Filname$
ASSIGN @Filname TO *
ASSIGN @Filname2 TO "file2.txt";FORMAT ON
ENTER @Filname2;Output$
ASSIGN @Cfile TO Filname$
```

Figure 5.3 : Interplay between the python and HTBasic code

So, now we can convert the binary data into meaningful data. The next step was to create a GUI which encapsulates these methods.

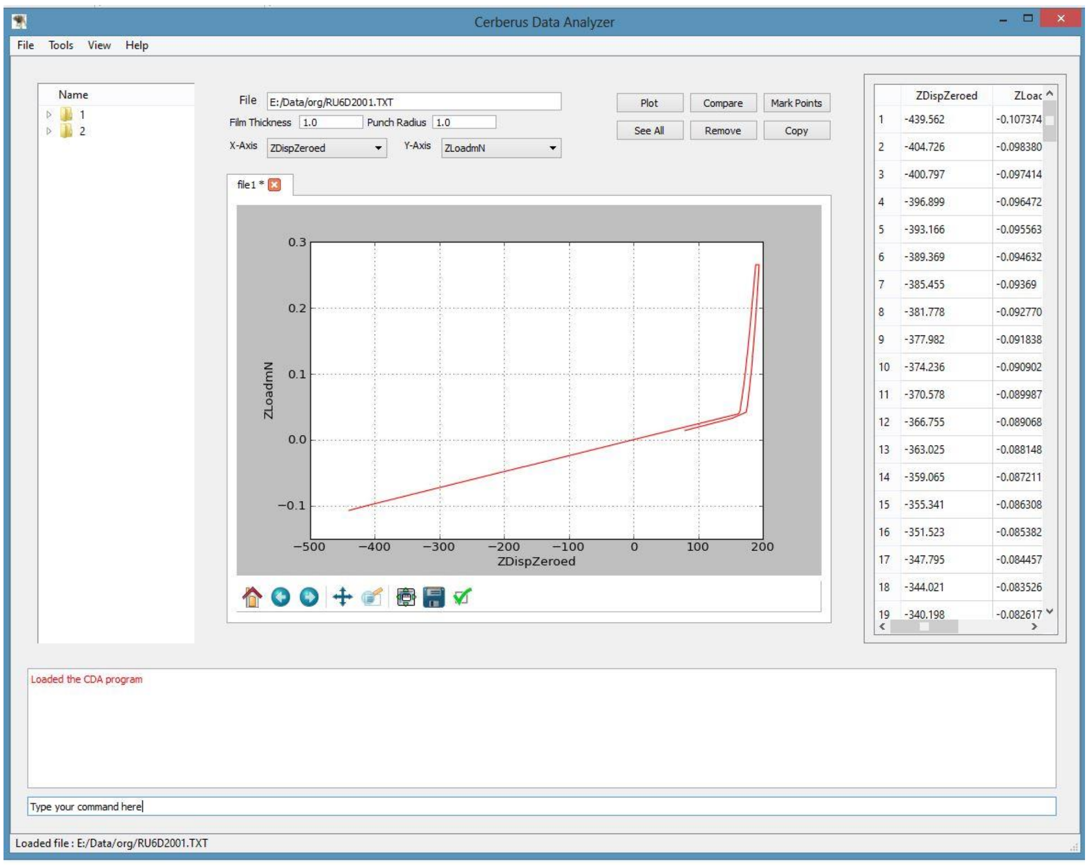
CHAPTER 6 – GRAPHICAL USER INTERFACE

A new GUI was designed to run on the windows system, which will have two main jobs-

1. To communicate with the old system.
2. To be able to modify the data and generate standard as well as custom plots.

6.1 GUI and the Framework

The following diagram shows the final GUI.



File E:/Data/org/RU6D2001.TXT
Film Thickness 1.0 Punch Radius 1.0
X-Axis ZDispZeroed Y-Axis ZLoadmN

	ZDispZeroed	ZLoac
1	-439.562	-0.107374
2	-404.726	-0.098380
3	-400.797	-0.097414
4	-396.899	-0.096472
5	-393.166	-0.095563
6	-389.369	-0.094632
7	-385.455	-0.09369
8	-381.778	-0.092770
9	-377.982	-0.091838
10	-374.236	-0.090902
11	-370.578	-0.089987
12	-366.755	-0.089068
13	-363.025	-0.088148
14	-359.065	-0.087211
15	-355.341	-0.086308
16	-351.523	-0.085382
17	-347.795	-0.084457
18	-344.021	-0.083526
19	-340.198	-0.082617

Loaded the CDA program

Type your command here

Loaded file : E:/Data/org/RU6D2001.TXT

Figure 6.1 : Cerberus Data Analyzer: Final GUI.

Python(x,y) was used as the framework to design the GUI. Various Python libraries were used. Some of them are as following-

(i) MatPlotLib- to draw the graphs and to provide tools to play with the graph.

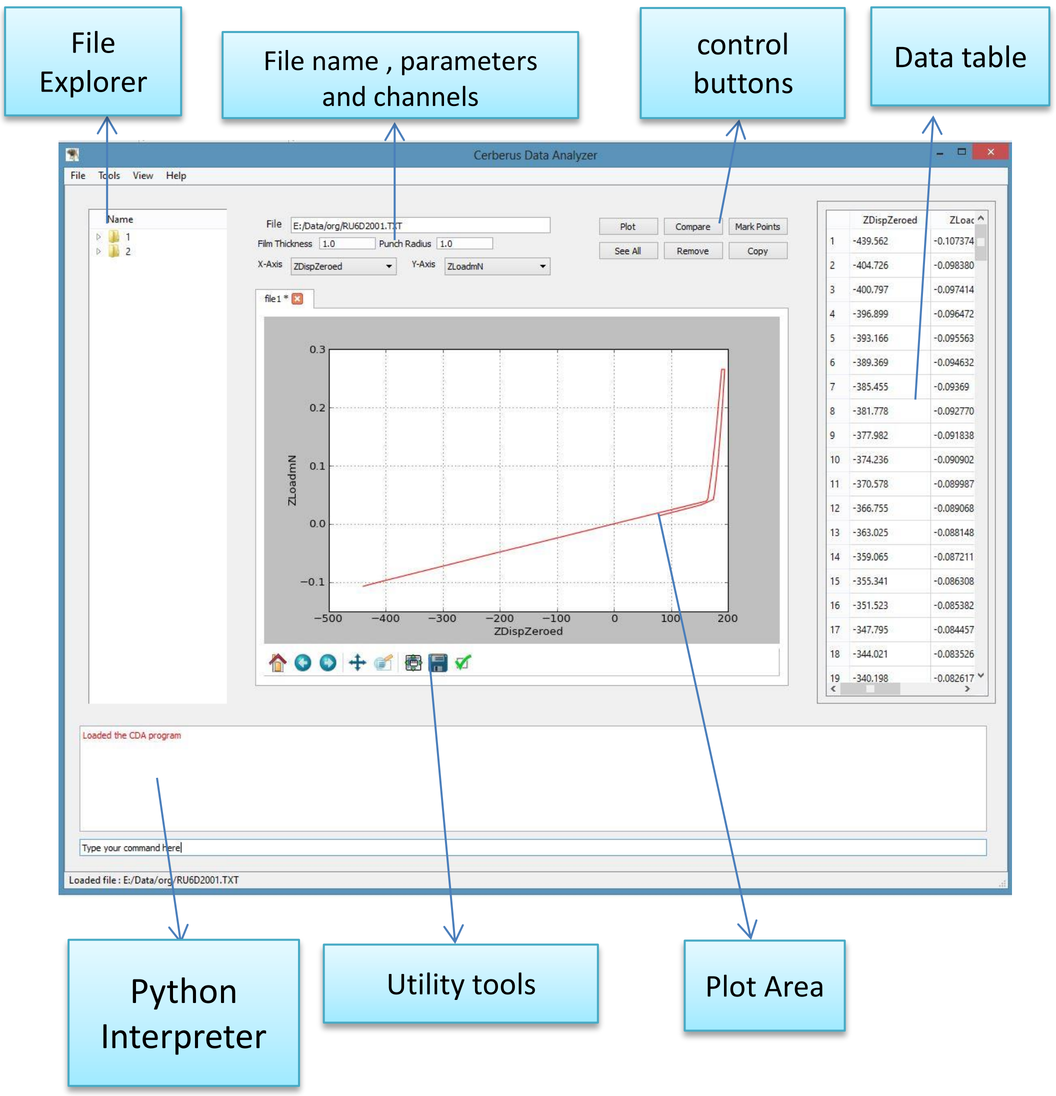
(ii) NumPy- for numerical computation purposes like Num array etc.

(iii) PyQt4- For GUI design and event handling.

(iv) pickle- to store an object and then access it again.

6.2 Description of the GUI

The following figure shows the different elements of the GUI.



File E:/Data/org/RU6D2001.TXT
Film Thickness 1.0
Punch Radius 1.0
X-Axis ZDispZeroed
Y-Axis ZLoadmN

	ZDispZeroed	ZLoac
1	-439.562	-0.107374
2	-404.726	-0.098380
3	-400.797	-0.097414
4	-396.899	-0.096472
5	-393.166	-0.095563
6	-389.369	-0.094632
7	-385.455	-0.09369
8	-381.778	-0.092770
9	-377.982	-0.091838
10	-374.236	-0.090902
11	-370.578	-0.089987
12	-366.755	-0.089068
13	-363.025	-0.088148
14	-359.065	-0.087211
15	-355.341	-0.086308
16	-351.523	-0.085382
17	-347.795	-0.084457
18	-344.021	-0.083526
19	-340.198	-0.082617

Loaded the CDA program

Type your command here

Loaded file : E:/Data/org/RU6D2001.TXT

Figure 6.2: Description of the GUI.

6.2.1 File Explorer

After discussion with the people who will be using the GUI, the need for a data explorer came to light. The data explorer should display all the calculated values in accessible way without creating a mess in the GUI.To achieve this goal, I used QTableWidget of PyQt which is designed for the systematic data view.

6.2.2 File name, Parameters and Channels

A line edit shows the address of the file opened in the current tab. Two line edits are also present to set the film thickness and punch radius.
Two combo box are their to set the x and y channels.

6.2.3 Data Table

After discussion with the people who will be using the GUI, the need for a data explorer came to light. The data explorer should display all the calculated values in accessible way without creating a mess in the GUI.To achieve this goal, I used QTableWidget of PyQt which is designed for the systematic data view.

6.2.4 Utility Tools

Various utility tools were added to help the user in the manipulating the results-

(i) Pan: Pan through the graph.
(ii) Back and forward: Undo and redo the changes made in the graph.
(iii) Home: Undo all the changes.
(iv) Zoom Rect: Zoom the specified rectangle.

6.3 Main features of the GUI

6.3.1 View

Using view we can see some standard curves by a click-

- Stiffness vs. load
- Displacement vs. amplitude
- Harmonic amplitude
- Phase angle vs. vertical displacement
- Cross Talk during approach

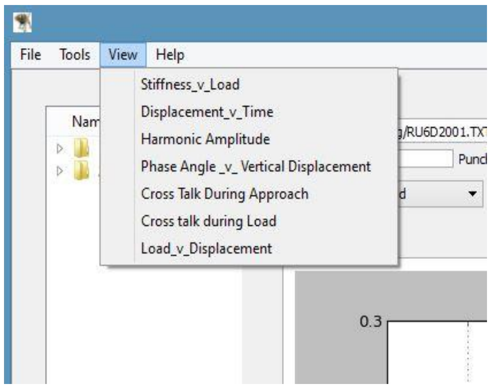
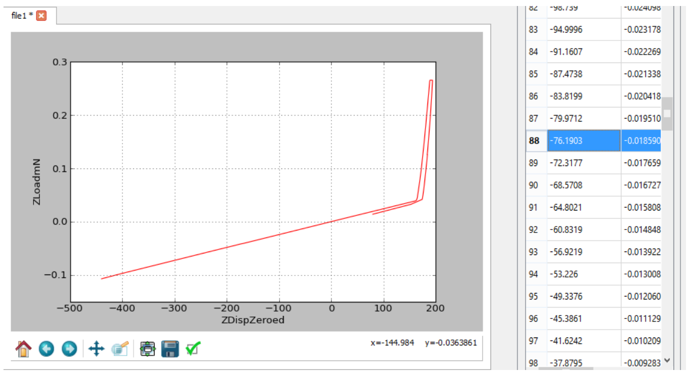


Figure 6.3: View menu: standard plots.

6.3.2 Mouse tracker

We may see the values at any point by clicking in the vicinity(+/- 5) of the Point, as the corressponding value gets highlighted.



82	-98.739	-0.024098
83	-94.9996	-0.023178
84	-91.1607	-0.022269
85	-87.4738	-0.021338
86	-83.8199	-0.020418
87	-79.9712	-0.019510
88	-76.1903	-0.018590
89	-72.3177	-0.017659
90	-68.5708	-0.016727
91	-64.8021	-0.015808
92	-60.8319	-0.014848
93	-56.9219	-0.013922
94	-53.226	-0.013008
95	-49.3376	-0.012060
96	-45.3861	-0.011129
97	-41.6242	-0.010209
98	-37.8795	-0.009283

Figure 6.4: When mouse is clicked near a point on the plot the corresponding value gets highlighted in data table.

6.3.3 Multiple Files

The task of opening multiple files is achieved through QTabWidget of PyQt. Each tab has its associated Dataset object which contains all the data and other details.

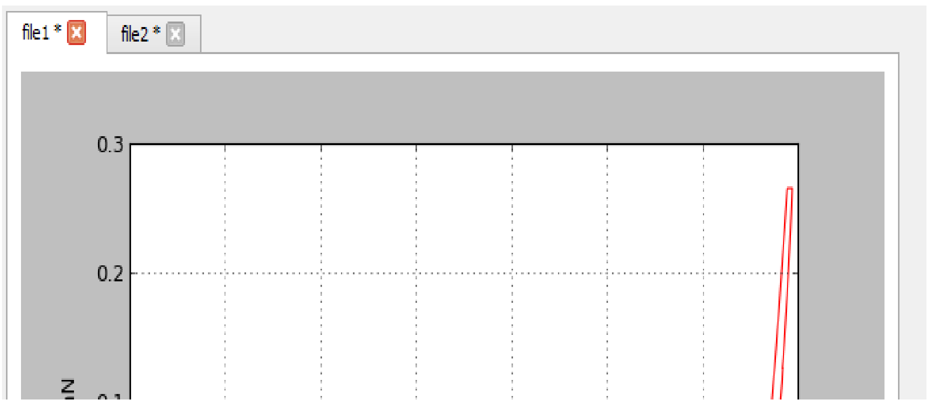


Figure 6.5: file 1 and file 2 are open at the same time.

In the next chapter, the main issue of the project is discussed, i.e., how the connection will be made between the two systems.

CHAPTER 7 – COMMUNICATION BETWEEN NEW AND OLD SYSTEM

After the completion of the GUI, the next target was to connect the GUI to the hardware. The initial strategy for connecting the GUI was through the Ethernet port but on more discussion and thinking another solution came to light- serial port (RS-232). So finally this option was selected and connection was established between the old control system and the modern GUI. But serial port is not a plug and play thing. We have to decide various parameters before the connection. The following sections deal with these details.

7.1 Serial (RS-232) Communication

A serial port is a serial communication physical interface through which information transfers in or out one bit at a time (in contrast to a parallel port). Throughout most of the history of personal computers, data was transferred through serial ports to devices such as modems, terminals and various peripherals.
There are three main issues regarding serial communication:

7.1.1 CABLING:

We have options of 9 pin or 25 pin serial communication.

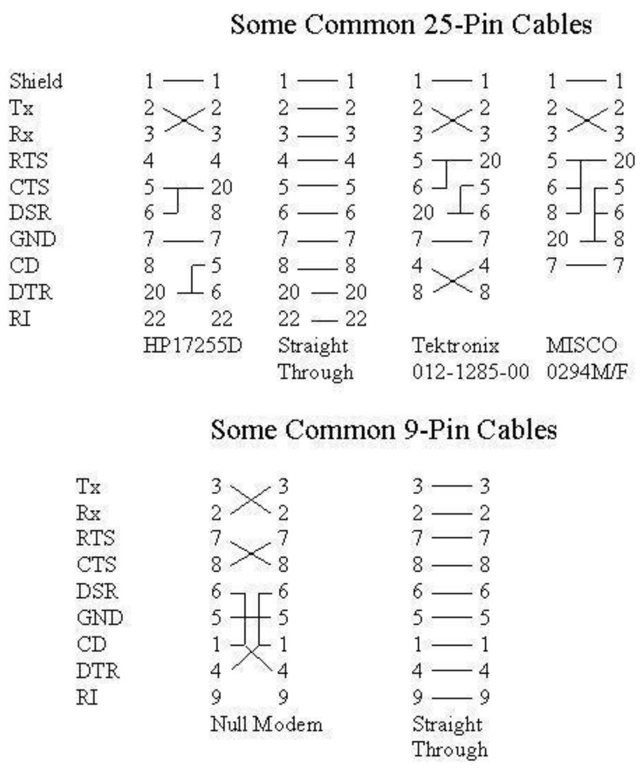


Figure 7.1: Serial Port cabling models.

7.1.2 HANDSHAKING:

There are two main handshaking protocols in serial communication:

XON/XOFF Handshaking:

When the buffer is about full, XOFF is sent.
When there is room again in the buffer, XON is sent.

Hardware Handshaking:

If the receive buffer is empty, turn on DTR and wait for a character to arrive. Get a character from the receive buffer and turn off DTR.

7.1.3 COMMUNICATION PARAMETERS:

Plugging the cable between the computer and the device is not all that is required to make them talk. The devices must speak the same language in order to understand each other. It doesn't matter what it is, but it is essential they be the same! The language consists of the baud rate, data bits, parity, and stop bits. We must determine which of these parameters can be set on the device, what they can be set to, and how they are set. Typically, you would choose the highest baud rate that both the device and the computer support, one stop bit, eight data bits, and no parity.

7.1.4 DATA FORMATS:

Once the computer and device are communicating correctly, it is still possible to get incorrect data if the data is formatted by the device in one way and interpreted by the computer in another or vice versa. The easiest way to get things working is to instruct the device to send data in ASCII, with CR/LF terminating each data item. However, the fastest way to exchange data is to do so in binary.

After lot of discussion and thought the final parameters were decided, which are as follows:

1. Use 9 pin serial interface.
2. XON/XOFF Handshaking.
3. Com Params

   Baud Rate: 9600
   Data Bits: 8
   Parity: None
   Stop Bits: 1

4. Since data in Cerberus is binary so binary data format is chosen.

7.2 FINAL CONNECTION

Figure 7.2 illustrates the connection model. In the figure, we may see that the whole system is divided into two parts. First , the windows system and second the dos system. The windows system consists of the main GUI, which when prompted to read from the stream evokes a HTBasic code which then communicates with the HTBasic on the second system via serial interface.

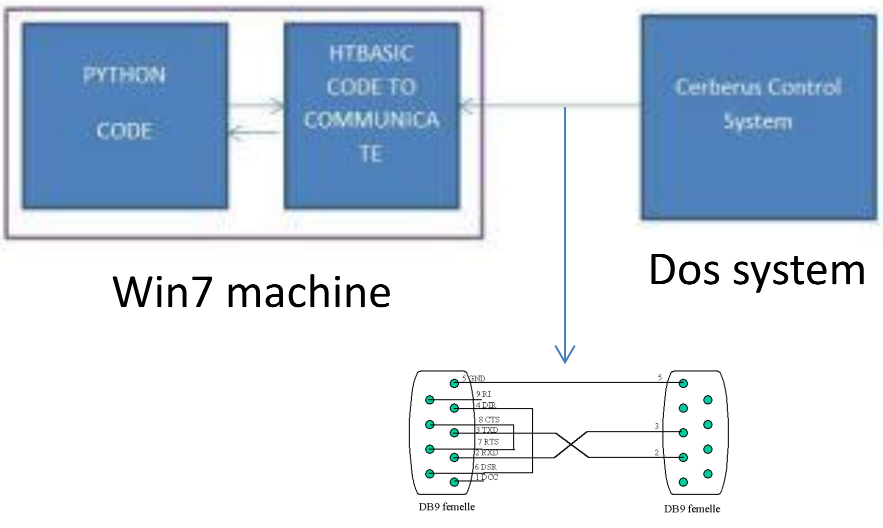
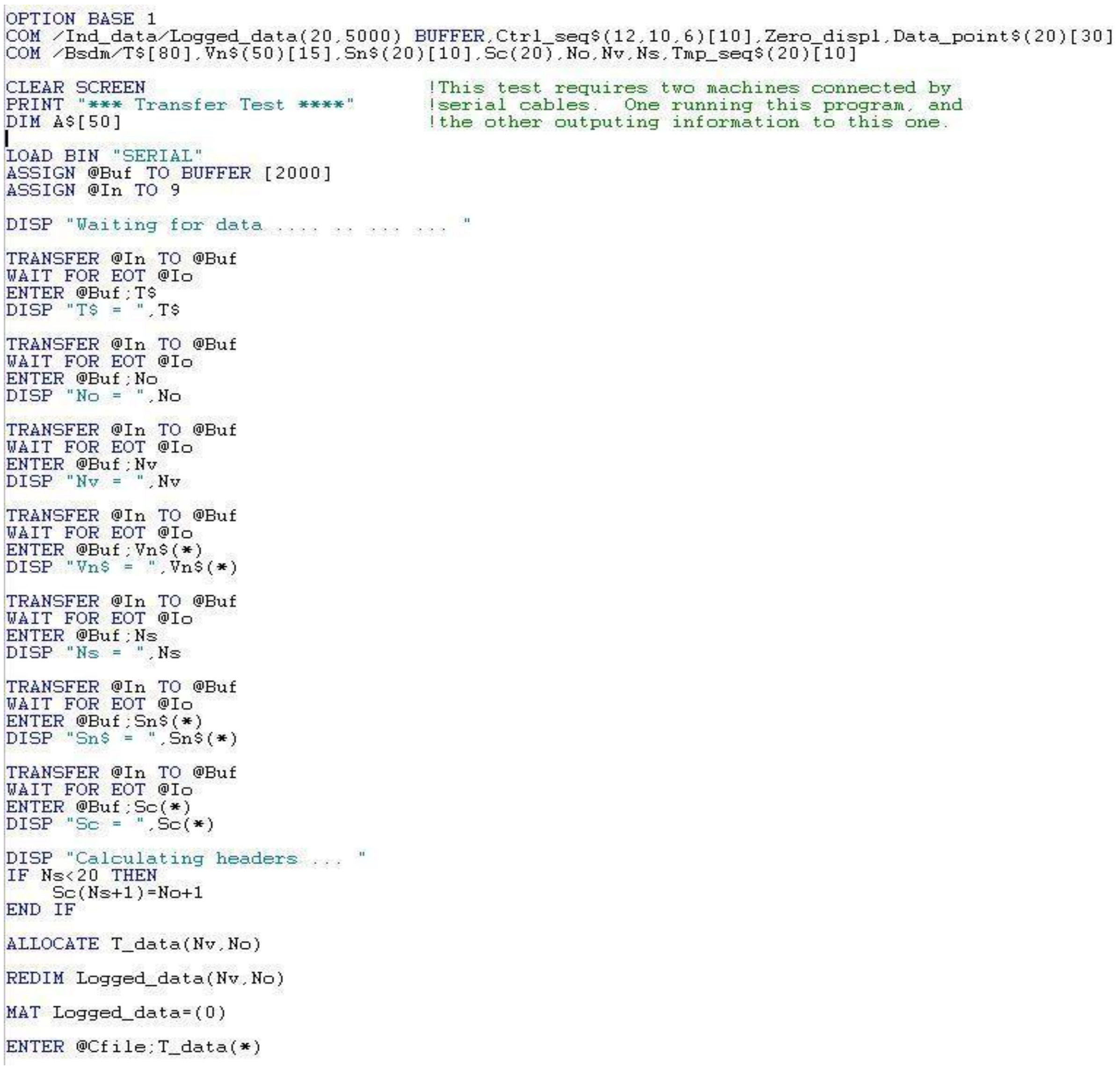


Figure 7.2: The whole model

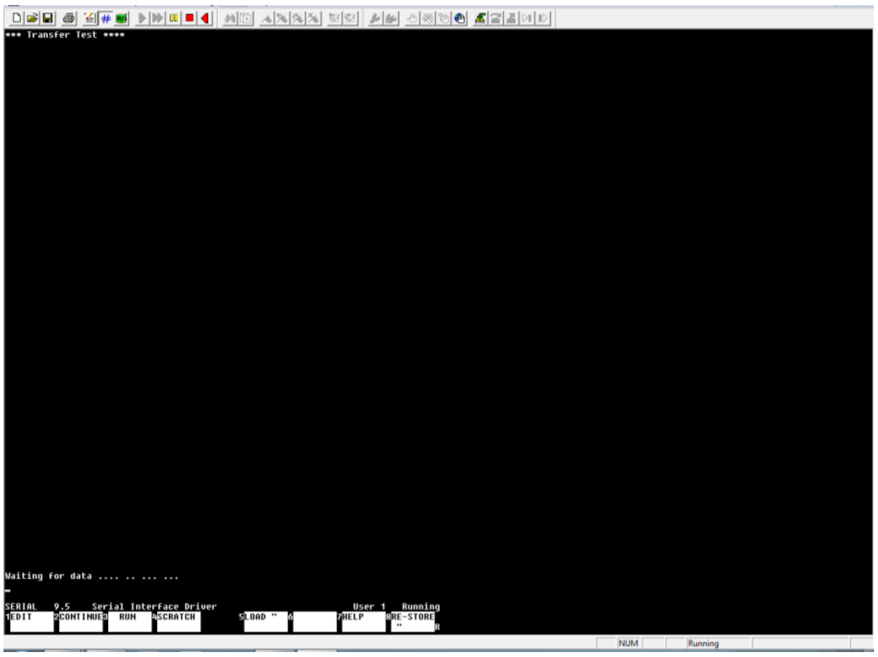
A part of HTBasic code to communicate is shown in the following figure:



```
OPTION BASE 1
COM /Ind_data/Logged_data(20,5000) BUFFER,Ctrl_seq$(12,10,6)[10],Zero_displ,Data_point$(20)[30]
COM /Bsdm/T$[80],Vn$(50)[15],Sn$(20)[10],Sc(20),No,Nv,Ns,Tmp_seq$(20)[10]

CLEAR SCREEN                           !This test requires two machines connected by
PRINT "*** Transfer Test ****"         !serial cables.  One running this program, and
DIM A$[50]                             !the other outputing information to this one.

LOAD BIN "SERIAL"
ASSIGN @Buf TO BUFFER [2000]
ASSIGN @In TO 9

DISP "Waiting for data .... .. ... ... "

TRANSFER @In TO @Buf
WAIT FOR EOT @Io
ENTER @Buf;T$
DISP "T$ = ",T$

TRANSFER @In TO @Buf
WAIT FOR EOT @Io
ENTER @Buf;No
DISP "No = ",No

TRANSFER @In TO @Buf
WAIT FOR EOT @Io
ENTER @Buf;Nv
DISP "Nv = ",Nv

TRANSFER @In TO @Buf
WAIT FOR EOT @Io
ENTER @Buf;Vn$(*)
DISP "Vn$ = ",Vn$(*)

TRANSFER @In TO @Buf
WAIT FOR EOT @Io
ENTER @Buf;Ns
DISP "Ns = ",Ns

TRANSFER @In TO @Buf
WAIT FOR EOT @Io
ENTER @Buf;Sn$(*)
DISP "Sn$ = ",Sn$(*)

TRANSFER @In TO @Buf
WAIT FOR EOT @Io
ENTER @Buf;Sc(*)
DISP "Sc = ",Sc(*)

DISP "Calculating headers ... "
IF Ns<20 THEN
    Sc(Ns+1)=No+1
END IF

ALLOCATE T_data(Nv,No)

REDIM Logged_data(Nv,No)

MAT Logged_data=(0)

ENTER @Cfile;T_data(*)
```

Figure 7.3: HTBasic code to communicate with the dos system.

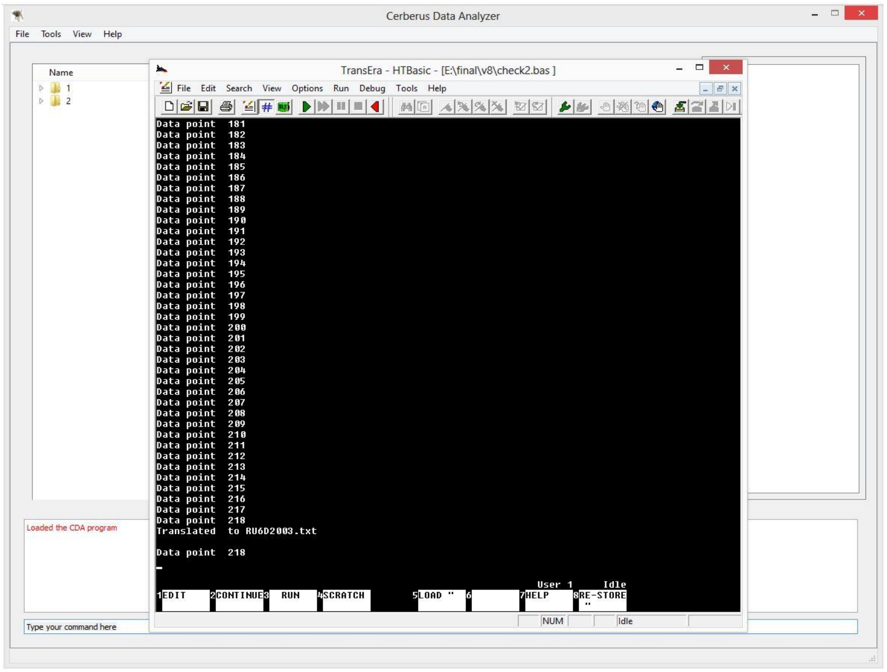
By using the model discussed in the previous section, the final connection was made between the two devices. The following figure shows the snapshot of the screen where HTBasic code running on the windows machine and waiting for the data.



```
*** Transfer Test ****










Waiting for data .... .. ... ...
_
SERIAL    9.5    Serial Interface Driver                    User 1   Running
1EDIT    2CONTINUE3  RUN    4SCRATCH       5LOAD "  6        7HELP    8RE-STORE
                                                                       "      R
```

Figure 8.1: HTBasic program on the windows machine waiting for the data.

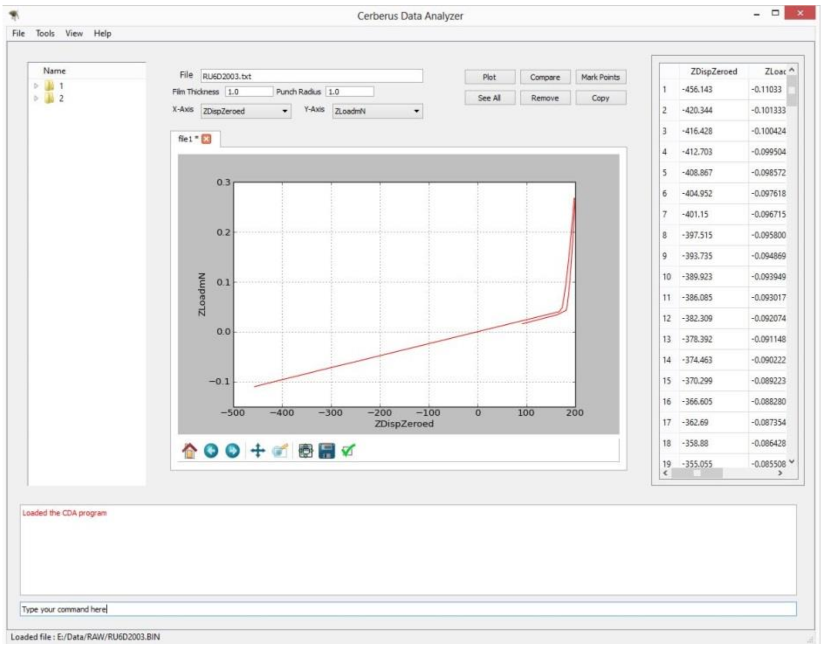
After the data starts coming, the HTbasic code shows the data points as they are received.



```
Data point  181
Data point  182
Data point  183
Data point  184
Data point  185
Data point  186
Data point  187
Data point  188
Data point  189
Data point  190
Data point  191
Data point  192
Data point  193
Data point  194
Data point  195
Data point  196
Data point  197
Data point  198
Data point  199
Data point  200
Data point  201
Data point  202
Data point  203
Data point  204
Data point  205
Data point  206
Data point  207
Data point  208
Data point  209
Data point  210
Data point  211
Data point  212
Data point  213
Data point  214
Data point  215
Data point  216
Data point  217
Data point  218
Translated  to RU6D2003.txt

Data point  218
_
                                            User 1     Idle
1EDIT    2CONTINUE3  RUN    4SCRATCH       5LOAD "  6        7HELP    8RE-STORE
                                                                       "
```

Figure 8.2: HTBasic code shows the data point getting received.

After all the data points are received the final load v disp plot is produced.



File RU6D2003.txt
Film Thickness 1.0 Punch Radius 1.0
X-Axis ZDispZeroed Y-Axis ZLoadmN

	ZDispZeroed	ZLoac
1	-456.143	-0.11033
2	-420.344	-0.101333
3	-416.428	-0.100424
4	-412.703	-0.099504
5	-408.867	-0.098572
6	-404.952	-0.097618
7	-401.15	-0.096715
8	-397.515	-0.095800
9	-393.735	-0.094869
10	-389.923	-0.093949
11	-386.085	-0.093017
12	-382.309	-0.092074
13	-378.392	-0.091148
14	-374.463	-0.090222
15	-370.299	-0.089223
16	-366.605	-0.088280
17	-362.69	-0.087354
18	-358.88	-0.086428
19	-355.055	-0.085508

```
Loaded the CDA program
```

Type your command here

Loaded file : E:/Data/RAW/RU6D2003.BIN

Figure 8.3: Final plot after all data points are received.

CHAPTER 9 – SUMMARY

The aim of this project was to develop a GUI interface that can interact with the existing control system in real time. I was able to complete this task. The medium used to communicate the data, although ancient, is very effective and the data transfer is very quick.

After this project, anyone who is going to use the Cerberus don't have to worry about the whole process to end, in order to see the result. Now, we can see the data live as it is coming. Also using the GUI he can play with various type of plots and can also play with the provided he has read the help manual of the project. The plots produced by the GUI are highly accurate and can be exported as LaTex images and can be directly used in research papers or articles.

The communication medium used is highly reliable and robust. It is one of most widely used communication channel. So, there is a little chance that the communication channel will be dysfunctional in a near future.

The best part of the solution that we adopted is that the processing and sequencing of the existing system does not get influenced by any of the changes. This is very important as the system running this system has very low processing capabilities. So, a adding one instruction means a lot. But we were able to do so, without causing much delay in the normal processing and a normal user experiences no change at all in the working of the system. So, an old user don't have adopt to the new system.

At last I would like to conclude by using a famous saying that the "**best solution is always the simplest one**". That's been the sole directive of this project. We were looking for the simplest possible solution to the problem and the end we found our solution in an outdated technology.

APPENDIX 1 – CALIBRATION CONSTANTS

		Z-Axis	X-Axis	Y-Axis
DC Calibrations				
	Displacement Constant Term	9944.3	18506	11091
	Displacement Linear Term	2.0903	4.8506	1.6121
	Displacement Squared Term	-0.3789	-0.5316	-0.26013
	Load Constant Term	47624	43313	58032
	Load Linear Term	8.0558	21.588	3.5987
	Load Squared Term	0.003	0.0067645	0.00064854
	Spring Constant Term	230	207	204
	Spring Constant Linear Term	0.0055042		
	Spring Constant Squared Term	-0.010195		
	Spring Constant Cubed Term	1.77E-05		
	Range 0 Gain	1.00E+00	1.00E+00	1.00E+00
	Range 1 Gain	2.00E+00	2.00E+00	2.00E+00
	Range 2 Gain	4.00E+00	4.00E+00	4.00E+00
	Range 3 Gain	8.00E+00	8.00E+00	8.00E+00
	Range 4 Gain	1.60E+01	1.60E+01	1.60E+01
	Range 5 Gain	3.20E+01	3.20E+01	3.20E+01
	Range 6 Gain	6.40E+01	6.40E+01	6.40E+01
	Range 7 Gain	1.28E+02	1.28E+02	1.28E+02
	DC Load Frame Stiffness	2.20E+05	2.20E+05	2.20E+05

AC Calibrations	Z-Axis	X-Axis	Y-Axis
Displacement Constant Term	9935.9	18509	11088
Displacement Linear Term	2.0639	4.8118	1.6513
Displacement Squared Term	-0.37801	-0.53142	-0.26013
Load Constant Term	14023.5	12767	17098
Load Linear Term	2.36	6.4062	1.1621
Load Squared Term	-0.0002	0.0019054	-0.0013593
Modcal Factor	1.15E-08	1.15E-08	1.15E-08
Mass of Indenter	1.65E-04	1.45E-04	1.40E-04
Damping Coefficient Constant Term	0.019	0.0066396	0.017396
Damping Coefficient Linear Term	2.61E-06	0.00E+00	2.61E-06
Damping Coefficient Squared Term	7.94E-06	0.00E+00	7.94E-06
Damping Coefficient Cubed Term	-3.46E-09	0.00E+00	-3.46E-09
Damping Coefficient Quad Term	3.10E-09	0.00E+00	3.10E-09
AC Load Frame Stiffness	1.30E+05	2.12E+04	2.58E+04
	1.86E+05	2.14E+04	2.81E+04
			2.69E+04

Individual Masses			
	Z-axis	X-axis	Y-axis
Mass from Uncoupled Dyncal	1.04E-04	9.96E-05	9.99E-05
Mass of Coupler	2.10E-05	2.10E-05	2.10E-05
Mass of Indenter (Ti rod)	1.47E-06	1.47E-06	1.47E-06
Mass of Quartz Rod	6.32E-06	6.32E-06	6.32E-06
	1.33E-04	1.28E-04	1.29E-04
Fitted Mass	1.65E-04	1.45E-04	1.40E-04

Area Function		Z-axis	X-Axis
	Lead Term	24.74350507	24.56
	Linear Term	738.3149628	272.89
	0.5 Term	-6111.09958	23376
	0.25 Term	7260.883126	-288280
	0.125 Term	4487.613794	754610
	0.06125 Term	831.9219679	-492830
	0.030625 Term	-1319.00673	
	0.0153125 Term	-2428.99083	
	0.00765625 Term	-2985.8581	

APPENDIX II – CALIBRATION VALUES

Z-Axis	
Displacement Range	[(ZDisp[1]/10000)-1] * Rounded to zero
Displacement Offset	((ZDisp[1]-((Z-Displacement Range+1)*10000))-5)/10 * Rounded to zero
DC Displacement Calibration	Z-DC- Displacement Constant Term + Z-DC- Displacement Linear Term * [Z-Displacement Offset-128] + Z- DC- Displacement Squared Term * [Z-Displacement Offset-128]^2
AC Displacement Calibration	Z-AC- Displacement Constant Term + Z-AC- Displacement Linear Term * [Displacement Offset-128] + Z- AC- Displacement Squared Term * [Z-Displacement Offset-128]^2
Load Range	[Zload[1]/10]-1 * Rounded to zero
DC Load Calibration	Z-DC- Load Constant Term + Z-DC- Load Linear Term * [Z-Displacement Offset-128] + Z-DC- Load Squared Term * [Z-Displacement Offset-128]^2
AC Load Calibration	Z-AC- Load Constant Term + Z-AC- Load Linear Term * [Z-Displacement Offset-128] + Z-AC- Load Squared Term * [Z-Displacement Offset-128]^2
Modcal Factor	Z-Axis Modcal Factor
X-Axis	
Displacement Range	[(XDisp[1]/10000)-1] * Rounded to zero
Displacement Offset	((XDisp[1]-((X-Displacement Range+1)*10000))-5)/10 * Rounded to zero
DC Displacement Calibration	X-DC- Displacement Constant Term + X-DC- Displacement Linear Term * [X-Displacement Offset-128] + X- DC- Displacement Squared Term * [X-Displacement Offset-128]^2
AC Displacement Calibration	X-AC- Displacement Constant Term + X-AC- Displacement Linear Term * [Displacement Offset-128] +X- AC- Displacement Squared Term * [X-Displacement Offset-128]^2
Load Range	[Xload[1]/10]-1 * Rounded to zero
DC Load Calibration	X-DC- Load Constant Term + X-DC- Load Linear Term * [X-Displacement Offset-128] + X-DC- Load Squared Term * [X-Displacement Offset-128]^2
AC Load Calibration	X-AC- Load Constant Term + X-AC- Load Linear Term * [X-Displacement Offset-128] + X-AC- Load Squared Term * [X-Displacement Offset-128]^2
Modcal Factor	X-Axis Modcal Factor
Y-Axis	
Displacement Range	[(YDisp[1]/10000)-1] * Rounded to zero
Displacement Offset	((YDisp[1]-((Y-Displacement Range+1)*10000))-5)/10 * Rounded to zero
DC Displacement Calibration	Y-DC- Displacement Constant Term + Y-DC- Displacement Linear Term * [Y-Displacement Offset-128] + Y- DC- Displacement Squared Term * [Y-Displacement Offset-128]^2
AC Displacement Calibration	Y-AC- Displacement Constant Term + Y-AC- Displacement Linear Term * [Displacement Offset-128] +Y- AC- Displacement Squared Term * [Y-Displacement Offset-128]^2
Load Range	[Yload[1]/10]-1 * Rounded to zero
DC Load Calibration	Y-DC- Load Constant Term + Y-DC- Load Linear Term * [Y-Displacement Offset-128] + Y-DC- Load Squared Term * [Y-Displacement Offset-128]^2
AC Load Calibration	Y-AC- Load Constant Term + Y-AC- Load Linear Term * [Y-Displacement Offset-128] + Y-AC- Load Squared Term * [Y-Displacement Offset-128]^2
Modcal Factor	Y-Axis Modcal Factor

REFERENCES

1. Poon, B; Rittel, D; Ravichandran, G (2008). "An analysis of nanoindentation in linearly elastic solids". International Journal of Solids and Structures 45 (24): 6018.doi:10.1016/j.ijsolstr.2008.07.021.
2. W.C. Oliver and G.M. Pharr (2011). "Measurement of hardness and elastic modulus by instrumented indentation: Advances in understanding and refinements to methodology". Journal of Materials Research 19: 3. doi:10.1557/jmr.2004.19.1.3.
3. “Dynamic proportional-integral-differential controller for high-speed atomic force microscopy”. Noriyuki Kodera, Mitsuru Sakashita, and Toshio Ando.
4. “Development of a depth controlling nanoindentation tester with subnanometer depth and submicro-newton load resolutions”. Atsushi Shimamotoa) and Kohichi Tanaka Department of Mechanical Engineering, Nagaoka University of Technology, Kamitomioka, Nagaoka,940-21 Japan.