Rui Pinto Guimarães

Application Software for an Instrumented Prosthesis for Knee Joint Force Measurement in Vivo
Application Software For An Instrumented Prosthesis For Knee Joint Force Measurement In Vivo
At
The Centre For Biomedical Engineering,
University College London


Rui Pinto Guimarães

Tutor at FEUP: Prof. Eugénio Oliveira
Tutor at BME - UCL: Dr. Stephen Taylor

November, 2003
To my father,
to whom I owe and I will never be able to return
Abstract

The experimental measurement of forces across the human knee joint in vivo is yet to be achieved. Data on forces applied across joints in the body are needed to know how to better design and test implants, define rehabilitation regimes, and provide basic data for other biomechanics studies.

The Centre for Biomedical Engineering, University College London has been researching a possible method using telemetry of data measured by strain gauges in an instrumented prosthesis. An inductive coil link powers the permanent implant and telemeters the data, which makes the system useful in the long term.

In this system, the strain data are radiotransmitted to a portable computer where they are analysed and the applied force system at each instant in time is determined from these data.

The goals of the traineeship were to develop an application to be used during future measurements to be carried out during activities of patients fitted with an implant; to modify previously developed software responsible for data communication between the computer and the implant and to study possible algorithms used to determine the applied forces using a Finite Element method.

The main application was required to manage data related to the patients and plot the forces across the knee in charts and using different perspectives allowing different analyses by the user.

Several technologies were discussed for the development of the application considering the project history and advantages/disadvantages for each requirement. From here the LabVIEW platform was chosen and its technological risks exposed.

After this analysis a development model was formed considering also the fact that LabVIEW uses the visual programming paradigm and its own computer language called G.

According to the model, three prototypes were developed, each one forming the basis of the following one, with the intention of eliminating the primary risks and progressively fulfilling the requirements.

The second goal of the traineeship, modification of the communication software, was also carried out in LabVIEW which in this case operates as a client to a C++ program via a DDE link.

Finally, the stress data from a Finite Element model of the implant was studied and from here the predicted strains for applied loads were deduced. These strain data represent a mathematical prediction of the real strain data that will be transmitted by the implant to the computer when forces are applied.

Combinations of the strains were studied whose sensitivities could be compiled in a matrix. The inverse matrix would yield the pattern of loads applied to the implant. Operating with this matrix it would be possible to determine the force system and its point of application.
Preface

This report is first written to describe the problems, choices and solutions occurring during this traineeship. The description given here is also a reference for future work in this project and a document that justify many of the chosen options. It can be used as reference for future developers.

For the general reader it can be interesting to note the analysis of different technologies for a relative large project. In software development, mainly the comparison between two different programming paradigms: the conventional textual programming and the pure visual programming.

For this particular project, the chosen was the visual programming and it has its own structure and model of development. As, at the moment, it is under discussion by the software community if this programming paradigm is efficient for large software applications (there is a general agreement that is good for beginners and educational purposes), the comments and work description included in this report can be useful for the followers of the discussion.

It is also interesting to observe the integration of various technologies, the contribution of different expertise with a range of different subjects taken from the software, electrotechnical and mechanical engineering.

Readers interested in this particular field of research, measurement of forces across joints in the human body, can find here a good description of the technological basis of the project.

For those readers more interested in the analyses of the traineeship itself it is intended to explain the chosen path given the requirements, obstacles and what was conquered during this formation period.
Acknowledgments

This traineeship would not be possible without the support of many individuals and institutions. First of all, I wish to thank to my tutors, Dr. Eugénio Oliveira at Faculdade de Engenharia da Universidade do Porto and Dr. Stephen Taylor at University College London for all their professional advice and also their friendly orientation that was of great importance.

My thanks to the UCL, the Stanmore Implants Worldwide and the Centre for Biomedical Engineering, and to all members of this department that always made me feel very welcome. Special thanks go to Annie Bartram, Josie Marshall and Jarmilla Dodd whose patience and kindness I still can't find the boundaries.

I also wish to thank all the involved persons from the Faculdade de Engenharia da Universidade do Porto. It feels good to go training to another country knowing that I still have a good backup at home.

Kind thanks to all my family, whose phone bill largely increased. Many thanks to all my supporting friends.

I finally would like to thank the financial support of the program PRODEP – Ministry of Education from Portugal.
Table of Contents

1 Introduction .................................................................................................................. 1
  1.1 The Training Institution ...................................................................................... 1
  1.2 Project Description ............................................................................................ 2
  1.3 Developed Products and Studies ........................................................................ 5
  1.4 Organization of the Documentation .................................................................... 7

2 System Analyses ......................................................................................................... 8
  2.1 Project Background ............................................................................................ 8
  2.2 Problem Description .......................................................................................... 10
  2.3 Goals and Requirements .................................................................................... 12
    2.3.1 Front-End Application ................................................................................ 12
    2.3.2 Communication Software Update ............................................................... 12
    2.3.3 Finite Element Analyses ............................................................................ 13
  2.4 Execution Plan ..................................................................................................... 13

3 Technology Analysis .................................................................................................. 15
  3.1 Technological Background ................................................................................ 15
    3.1.1 General Project ......................................................................................... 15
    3.1.2 Communication System ............................................................................ 16
    3.1.3 Algorithm Development For Definition of Forces In The Knee Joint........ 17
  3.2 Technological Options ......................................................................................... 17
  3.3 Technological Risks ............................................................................................. 19

4 Development Model of the Front End Application .................................................... 21

5 System Architecture .................................................................................................. 23
  5.1 Architecture principles ....................................................................................... 23
  5.2 First Prototype ..................................................................................................... 24
  5.3 Second Prototype ................................................................................................. 25
  5.4 Third Prototype .................................................................................................... 29
  5.5 Conclusion ........................................................................................................... 31

6 Development Description .......................................................................................... 32
  6.1 Prototype Development ....................................................................................... 32
  6.2 Description of the Final Prototype ...................................................................... 33
    6.2.1 Structure Overview ..................................................................................... 34
    6.2.2 Physical Description ................................................................................... 34
    6.2.3 Visual Interface ............................................................................................ 35
  6.3 Conclusion ............................................................................................................ 42

7 Modification of the Communication System .............................................................. 43
  7.1 Data Decoding ..................................................................................................... 43
  7.2 Software Modification ......................................................................................... 44
  7.3 Conclusion ............................................................................................................ 46

8 Algorithm Development Based in FEA ................................................................... 47
  8.1 Calculation of Strains ......................................................................................... 48
  8.2 Matrix Method ..................................................................................................... 51
  8.3 Conclusion ............................................................................................................ 52
9 Conclusions and Future Work ................................................................. 53
References and Bibliography ................................................................. 55
APPENDIX A: Original List of Requirements ........................................... 56
APPENDIX B: Principal Stresses ............................................................ 59
APPENDIX C: Strain Gauges Basics ......................................................... 62
Table of Figures

Figure 1 - Knee replacement - arthroplasty ................................................................. 2
Figure 2 - Tibial tray with instrumentation and induction coil ..................................... 3
Figure 3 - Patient with power supply (inductive coil) for the implant ......................... 3
Figure 5 - Dr. Stephen Taylor operating the calibration rig ......................................... 5
Figure 6 - The knee anatomy ......................................................................................... 8
Figure 7 - Example of the result stresses (Von-Mises results) of a finite element analysis
applied to a model of compartment of the implant when an axial force is applied ......... 9
Figure 8 - Strain Gauges inserted in an implant ............................................................ 9
Figure 9 - Scheme of the calibration rig ....................................................................... 10
Figure 10 - The overall system with the used technologies ........................................... 11
Figure 11 - LabVIEW code for an example state machine ........................................... 22
Figure 13 - Use case for the second prototype ............................................................. 27
Figure 14 - Class diagram for a new prototype ......................................................... 28
Figure 18 - The Physical Model corresponding to the Logical Level of the final prototype... 34
Figure 19 - UI 5 - The complete physical model ............................................................ 35
Figure 20 - Visual Interface – Implant Data ................................................................. 36
Figure 21 - Menu System ............................................................................................ 37
Figure 22 - New functions for the Admin (Super user) .................................................. 37
Figure 23 - Opening a file with mouse click in a table cell corresponding to a session ...... 38
Figure 24 - Implant Data – View of the user interface ................................................... 39
Figure 25 - User Interface – Advanced Graphical View ................................................. 41
Figure 26 - Table showing the auxiliary bits sent per each frame .................................. 43
Figure 27 - Vi’s of the client application for the Mk4 program ........................................ 45
Figure 28 - Physical model of the serial port reader ..................................................... 46
Figure 29 - Finite Element implant model – Axial load applied .................................... 47
Figure 30 - Axis system for each half bridge of strain gauges ....................................... 48
Figure 31 - Formulas for calculating new stresses when there is theta rotation ............ 49
Figure 32 - Comparison between results of new formula, old formula, and calibration rig data .................................................................................................................. 50
1 Introduction

This traineeship was conducted at the Centre for Biomedical Engineering, University College London. Here, the work done contributed to the advance of leading research in measurement of forces across the human knee joint by radio telemetry using strain gauged knee implants. Next is described the Institution where this traineeship occurred, the project that was being carried out, the requirements, resulting product and how this report documents it.

1.1 The Training Institution

University College London (UCL) was founded in 1826 with strong social principles that challenged the existent discrimination of that age. A broaden vision gave UCL a long tradition in accepting international students that proudly maintains. Students from more than 130 countries outside the UK study at UCL, making up almost a third of the student body. Its community of 24,500 staff and students is engaged in productive partnerships around the world. Its excellence provided 18 Nobel Prizes to academics and graduates.

From UCL, the Institute of Orthopaedics & Musculo-Skeletal Science, is closely associated, and is on the site of, the Royal National Orthopaedic Hospital Trust. The reason for which the Institute was created is clear:

"Mission Statement: Creation of an international Centre of Excellence advancing musculo-skeletal science for the prevention, treatment, management and rehabilitation of musculo-skeletal disease. Making Science work for Patients"

These are their own words whose meaning is visible in their activities. To pursue this vision the Institute comprises four centres, under the overall leadership of Prof Allen Goodship.

- Clinical Orthopaedics, Acting Head, Mr John Skinner
- Biomedical Engineering, Head, Professor Gordon Blunn;
- Centre for Human Performance, Head, Professor Martin Ferguson-Pell;
- Musculo-Skeletal Pathobiology, Head, Professor Allen Goodship.

The main research of one of these centres, the Biomedical Engineering Department, where the traineeship occurred, is the study and improvement of implants. In bone tumour implants, The Centre has investigated different ways of fixing these implants to the bone. Over the last 5 years, all of these have been investigated experimentally and new methods of fixation are being used in clinical practice. The research programme in the fixation of standard total joint arthroplasty has expanded in relation to the requirements in expertise in different disciplines. For example biomaterials, telemetry and implantable motors for "growing prostheses" used in children, cell biology (including use of stem cells), specialised
morphometry and histopathology. Assessment of functionality of the improved implants are linked to the recent inclusion of the comparative biomechanics of movement unit in the Centre for Human Performance. The excellence of the research is well visible in the success of Stanmore Implants Worldwide Ltd., a spin-out of UCL, with which there is a close collaboration.

1.2 Project Description

Data on forces applied across joints in the body are needed to know how to better design and test implants, define rehabilitation regimes, and provide basic data for other biomechanics studies.

Although forces acting across the hip joint are now well documented both by analysis and in vivo measurement, those acting at the knee have so far only been determined analytically using a combination of mathematical models, kinematic data and ground reaction forces. These methods are subject to many simplifications and assumptions. The aim of the present work is to measure the forces acting across the femero-tibial joint of a permanently implanted total knee replacement in vivo.

Figure 1 - Knee replacement - arthroplasty
Forces can be measured directly during a variety of activities using instrumented implants, in which subjects are supplied with implants modified to enclose strain gauges and signal conditioning circuitry.

![Figure 2 - Tibial tray with instrumentation and induction coil](image)

A total of 24 strain gauges (top-left part of the figure) are arranged around the periphery and under each compartment. Wired in 12 half bridges, they measure the strain that is then amplified, digitised and telemetered.

![Figure 3 - Patient with power supply (inductive coil) for the implant](image)

The implant coil is the only part of the implant to be housed outside the hermetic
enclosure, and is connected to it via a metal-to-glass seal. The serial data is telemetered by impedance modulation of the inductive link, and the signal is detected at the external inductive energiser. This design requires only one implant and one external coil, which is placed around the tibia (see figure 3) during measurements, connected to the lightweight battery-powered inductive energiser. The serial signal is radio transmitted from the subject and received in real time by a portable PC used in routine measurement sessions, and can be processed off-line if necessary. Thus there are no wires trailing from the subject, giving full freedom of movement.

With the computer, considering the measured strain in the implant, it is possible to determine the axial force, the antero-posterior and medio-lateral shear forces, and the X and Y locations of the centre of applied load on the tibial tray.

![Diagram showing measured parameters on each compartment of tibial tray](image)

**Figure 4 - Measured parameters on each compartment of tibial tray (superior view): axial force and torque, AP and ML shear forces, AP and ML co-ordinates of load centre**

For tests and to experimentally determine the response of the gauges to each applied force and moment in each implant a calibration rig was built.

This complex project, entitled “AN INSTRUMENTED PROSTHESIS FOR KNEE JOINT FORCE MEASUREMENT IN VIVO”, is internationally considered of great importance. It is supposed,

- To expand the knowledge base of musculo-skeletal biomechanics, with particular relevance to the fields of Orthopaedics, Sports Medicine, and Human Performance.
- To specify the service and testing loads for total knee replacements, artificial ligaments, autografts, allografts and tissue engineered structures.
- To specify the applied loads for knee simulating machines for wear testing.
- To define the most appropriate rehabilitation regimes following knee injury and surgical procedure. Also, to evaluate different footwear, knee braces, insoles etc.
• To determine the functional behaviour of Total Condylar Knee Replacements, and thus to identify improvements in knee design.

• To provide force data at the knee for validation of knee models in the literature.

• To identify activities which produce excessive forces likely to prove detrimental to the prosthesis survival or surrounding tissues in the short or longer term.

![Figure 5 - Dr. Stephen Taylor operating the calibration rig](image)

At the moment this report is being written there are four engineers working for this project under the guidance of Dr. Taylor. Three under the hardware design and one in the study of software algorithms using the calibration rig. It is the conjunction of their expertise in electrotechnical, mechanical and industrial engineering that makes the project to progress.

1.3 Developed Products and Studies

For the referred project it was developed an application to be used in a laptop during the activities of the patients. The application was required to save data about the patient, maintain a complete record of the activities carried out and graphically present the results of the measurements.

Once analyzed several possible technologies, balanced advantages and disadvantages according to the requirements of the application and considering the technologies already used in the project, the chosen was LabVIEW (Laboratory Virtual Instrument Engineering Workbench). It is a platform that is provided with a range of tools to create applications to
support laboratory activities. It uses the visual programming paradigm including its own computer language, the G language, and provides its own development environment with an automatic compilation before the code execution.

This platform, with a different programming paradigm from that used in conventional textual languages, presents its own problems and virtues.

For the user interface layer it does not provide great flexibility of development once it has its own visual components for each software object that comes with the platform and from which the application is built, although they are customizable to a certain degree. For the logical layer it is limited in its data structures and does not provide the same effectiveness like the conventional textual languages. This, between other issues, makes the created application to waste computer resources.

Nevertheless, it provides a friendly development environment (intimately connected with the G language), a good variety of drivers and interfaces to communicate with other technologies and hardware and software components prepared to laboratory measurements and equipment.

The created application fulfills the functional requirements having some problems with overall performance. Every new data in the application would have to be plotted in ten milliseconds for a real time data display. Tests show that the problem is not with CPU processing power or main memory. New tests will verify if upgrading the hardware for graphics improves the speed of the data plotting to the required level.

The second goal of the traineeship was to modify the software, due to protocol changes, responsible by the computer data acquisition through the serial port. This software uses a program written in C++ to receive the data and deliver the information as a string to be used by a LabVIEW application. After some modifications to the LabVIEW program, an error was discovered in the simulated data source. This part is now on standby until the person responsible for the creation of the data source solves the problem. That did not happen during the time predicted for this traineeship.

Finally, the third goal was to study possible algorithms to be integrated in the created application during the achievement of the first goal. How this integration will be done is an issue to be studied later in project. This study tried to define the applied load in the implant using the resulted values of the inserted strain gauges.

For this study it was used data from a Finite Element Analysis made to a model of the implant. This had the advantage, compared with calibration rig method, of not exposing the results to mechanical and electrical problems. Although, by this method, only a small set of data was available.

The study was done with Microsoft Excel and an algorithm was built using the functions that this tool provides. The result was an iterative procedure based in a matrix, compiled from the strains values caused by the applied load chosen from defined list and that is defined as a guess. Multiplying this matrix by the vector that defines the actual load returns the error that the guess has when compared with true values of that load. The procedure tries to decrease the error to a maximum making a better guess in each iteration.

After developing the procedure it was noticed that although there were promising results, the available amount of data used was not enough to make final conclusions and
evaluate if it was or not possible to use the matrix procedure in the final application.

1.4 Organization of the Documentation

The documentation skeletal is provided by the Faculdade de Engenharia da Universidade do Porto for the trainees in Informatics and Computer Engineering and it was adapted to best describe this traineeship.

After this introduction that describes the circumstances of the traineeship and the achieved work, it follows, in the second chapter, a detailed description of the goals and how they emerged from project.

Possible technologies to be used for the traineeship are analyzed in the third chapter. Technologies that are related to project in some way, because they were used in similar projects or because they were used in previous phases of this project, are presented. Then, considering the requirements of the problem, the possible technologies to be used are discussed. From that the chosen ones are presented and possible risks from their use exposed.

The fourth chapter considers the circumstances of the traineeship, software requirements and technology risks to present a model of development of a software application. How this model was used in practice is then described in chapters number five and six.

The fifth chapter describes the conceptual analyses and specifications used during the development of the required application. The application is conceptually organized in different modules according to the requirements and UML diagrams are used for a better description of the specifications used.

The physical model of the application and its relation with the specification previously presented is described in the sixth chapter.

The seventh chapter presents the work done for the second goal. Because it does not require the development of a complete application but some modifications and additions to what was already done, it starts to present the start situation of the system. It follows the presentation of the changes it required and finally what was reached how it was reached.

The third and last goal of the traineeship is presented in the eighth chapter. It is focused on a study of a possible algorithm to be used in the application developed as the first goal. Its approach is mainly theoretical and tries to describe the ideas behind its development.

The final chapter makes the conclusions of the traineeship and indicates the future work to be developed.

It follows the bibliography written according the Chicago Manual of Style.

In appendix A is possible to find the original list of requirements for the main description as written in the start of the traineeship. The two other appendixes provide basic information on strain, stresses and strain gauges.
2 System Analyses

In the knee there are potentially 6 degrees of freedom per condyle (the contact part of the bone in the joint; see figure 6). The described project that is being carried at the Centre for Biomedical Engineering intends to measure the applied force by the femur to the tibia that is the resultant of three forces: an axial force and the two shear forces in perpendicular directions. It also intends to measure the point of application of the resultant force that is defined by two coordinates and, if possible, a torque produced by a rotation on the knee. To achieve this goal an implant and a complex system of instrumentation and software is being developed.

During previous phases of this project tools were used and software was developed that necessarily influenced the choices being made during the training period. In this chapter is described the goals to achieve during the training and how they emerged from the project.

![Figure 6 - The knee anatomy](image)

2.1 Project Background

This project finds itself in fairly advanced phase. In a first phase the purpose was to find a design for the implant that had to pass fatigue tests and be accepted by the surgeons. After the choice of the design it was necessary to build instrumentation to be inserted and include the strain gauges. The position of the strain gauges in the implant was chosen with the help of a Finite Element Analysis applied to an implant model (Figure 7).
Figure 7 - Example of the result stresses (on the right – Von-Mises results) of a finite element analysis applied to a model of compartment of the implant when an axial force (on the left) is applied

With these results was possible to visualize the points of greater stress giving example load cases and choose not only the location but also the orientation of the gauges. The chosen orientation was intended to capture the principal stresses (for more information about stresses consult appendix B) at the location point of the gauge and improve the sensitiveness of the gauges to applied loads. This had a major importance because, due to the compartment’s complex geometry, the variation of strain with respect to load position is nonlinear. To discover the relationship between them, and predict the behavior to the applied load, a calibration rig was then built, in a second phase.

Figure 8 – Strain Gauges inserted in an implant

With this machine, using an implant with strain gauges inserted, it was possible to experimentally apply a load and display in a computer the measured strain. A set of combinations of position, magnitude and direction of force can be pre-programmed in the computer that then controls the calibration rig and automatically executes the instructions. The results of the applied loads are measured, sent to the computer through the serial port and saved in memory.
For this phase, occurring during this training period, it would be necessary to find the algorithms that will be used to determine the forces and the position of its application on the tibial tray of the implant. These algorithms will be used in a computer that receives the data sent by the implant by radiotransmition. The hardware and software that support the communication are also to be improved and the application that will display and manage data sent from the implant is to be developed.

2.2 Problem Description

Under the guidance of Dr. Taylor, others students have worked in the project building the calibration rig and the software that control it and receive the data. The software to support the communication, in the calibration rig, was developed with the computer language C but the data presentation and treatment was developed using the LabVIEW Development Environment from National Instruments.
The calibration rig is wired to two serial ports of the computer from which it receives the data. Data is acquired from the acting indenter (see figure 9) mainly for error compensation. Data is also acquired from the tibia prosthesis, which will be the crucial data to determine the final variables: forces and location of application. This data is passed to a LabVIEW program to be displayed in a comprehensive way. To give more flexibility to LabVIEW, its creators tried to take the most from the Microsoft Windows supported technologies. At the time the software for this project was created, LabVIEW was supporting DDE (now supplanted by OLE) as the main technology for interprocess communication, which was used in this case. The PCI-DIO48H board is add to the computer and permits the control of the hardware of the calibration rig. LabVIEW uses a library offered by the manufacturer of this board and permits its interactivity.

Once the prosthesis is inserted in the patient it will send the data to the computer by radio transmission. Some of the work already done with the calibration rig may and should be reused. A goal of this training period was to change the communication software due to new requirements to be used with the prosthesis inserted in the patient.

Another goal was to complete the software system with the application that will receive the data in the computer, display it according to several options, and store it in a proper format for later availability. The algorithms to determine the forces and location of application were to be developed by another member of the department and then to be programmed and integrated in the application.

Due to the complex geometry of the tibia prosthesis, causing nonlinearity between the measured sensitivities in the strain gauges and the location of the applied force, the pretended algorithms were not easy to develop. Taking this into consideration, it became a goal, later during the training, to study alternative algorithms and ideas using a Finite Element Analysis. Unlike the measurements from the calibration rig, the FEA is not subject to such errors as electromagnetic interference, electronic or strain gauge drift, or mechanical tolerances. It is
then suitable for theoretical validation of algorithms and for rapidly testing new ideas. But large sets of data, as are given by the calibration rig, are not at disposal and some others solutions are only feasible to find with the practical experience.

The main goal of the use of the FEA was to test a previous idea with a more actualized model and therefore an all new set of data. Representing 'ideal' data, it could be used to show the likely success or failure of a specific method for determining the applied forces.

2.3 Goals and Requirements

2.3.1 Front-End Application

1 To integrate the data interpretation algorithms (project of another member of the research team) into a host environment for real-time data acquisition and interpretation. It should be possible to select calibration data for any one of several implants from a text file, for data logging.

2 Write a user interface to log and plot the data with context-sensitive filenames in real time.

3 Graph plots will be needed to optionally view the results of one or more channels, in real time, and from saved data. Each acquisition will require a filename with fixed format having the patient’s number, the activity code letter, and the session number.

4 Decode the timing of an external event to enable synchronisation with kinematic data from a motion analysis system.

2.3.2 Communication Software Update

The Mk4.exe program, written in C++, acquires serial data at 38,400 baud and decodes the data into the 12 LEFT and 12 RIGHT channels (strain gauge half bridges) from the implant. Auxiliary data also encoded are:

- Temperatures in the compartments (5 temperatures in each compartment)
- Synchronisation data (required by the Mk4 program for decoding the data)
- MODE of data transmission
- GAIN of the implant amplifier
- OVERSHOOT/UNDERSHOOT event in Left or Right compartment.
The mk4.exe program takes account of the MODE and GAIN in presenting the data.

This program will have to consider additional data that are also now encoded in the implant signal:

- Patient/prosthesis number (encoded in the implant)
- Patient leg, Left or Right (encoded in the implant)
- Patient box battery voltage (lithium ion battery)
- Patient box SMPSU energiser voltage (set by remote generator)
- External event indicator (event at the patient)

2.3.3 Finite Element Analyses

A finite element model of one compartment of the revised compartment design was loaded with 43 load cases representing change in each of the forces and positions to be determined. It is proposed to analyse these results to establish a method of iterative linear analysis for determining the forces and positions based on the available strain data. The results of this analysis should also be compared with the results from loading an instrumented component in the calibration rig, for establishing the method. Other strategies (apart from iteration, e.g. neural networks) can be tried to assess the likelihood of success.

The advantage of using FE data for this is that it will not be subject to experimental errors, and thus can be used to gain insight into the behaviour of the mechanical structure. The disadvantage is that the model does not accurately represent the actual implant, and so physical calibration of each implant is still needed.

2.4 Execution Plan

Once the three main tasks could be done concurrently, each one deserves its own schedule. There must exist a flexibility strategy that allows changing task when it is needed. That is, if the continuation of a task has to stop by any external reason, it is possible to continue with other one with no lost of time.

The main task is the software application described in the first goal. Its study and design should be the very beginning of the traineeship.

The update of the software in the second goal can be done later and at any moment. It is also dependent of the advances of other engineer working in the project that is developing the hardware that will be the data source for the program. This task should not take more than two weeks.
The third goal is the continuation of previous ideas and the integration of new ones in a theoretical study. It should be done once the main application is finish or near that.
3 Technology Analysis

The analysis of the technologies to be used during this traineeship is of major importance. It will define more precisely the amount of work to be done and what should be the best approach to develop the software.

This analysis considers the technologies used in a similar project and the ones already used in this project. It considers the requirements for the application of the first goal of the traineeship and considers vantages and disadvantages of usage.

The chosen platform was LabVIEW (Laboratory Virtual Instrument Engineering Workbench) that besides being in use in the project it simplifies the development of some of the requirements.

3.1 Technological Background

One project, well know in this research area, from the University of Berlin, about the measurement of forces across the hip is here referred. It is seen as the gold standard for telemetry in total joints. Though, as the project was conducted before the 90's, quickly one concludes that its technologies are somewhat outdated as the information technology developed much faster that this particular area of Biomechanics.

The technologies used in previous phases of this project are also here presented.

3.1.1 General Project

The University of Strathclyde [MOR68], [MOR69] was pioneer in the analytical study of the forces across the knee joint [HER00]. Combining analysis of the way we walk and the biomechanical models it gave a starting point for predicting and measuring these forces.

Although it is a noninvasive and relatively cheap technique there are many suppositions on its analysis. For an instance, no physiological effects on the knee were considered. There is no analysis for the muscles actions on the knee and their different combination effect. For more accurate results the measurement in vivo became important.

Although forces acting across the hip joint are now well documented by in vivo measurement those acting at the knee have been revealed to be hard to measure due to the joint irregularities. As there is at the moment no accurately successful measure, data, using this method, would be well received internationally with diverse practical applications.

Telemetry of strain gauge data was also used for the hip analysis process [BER90]. The method is similar in the part responsible for the power supply. The differences occur mainly in the way the data is captured, telemetered and transformed to the final form, to give the forces across the joint.

The authors name their method to obtain the forces from the strain measurement “The matrix method”. It is based in the algebraic manipulation of the vectors that define the relevant data, being a starting point, the vector of the signals generated by the three strain
gauges inserted.

The technique was presented in 1988 [BER88] and the procedure is still maintained. More recent projects use the same method and only slight changes were included for more control on the accuracy of the data being the mathematical method unaltered [BER99].

The success of this work had a natural influence in the way this knee project has been built. One of the first ideas, to study with Finite Element Analysis, will be "matrix" based. Naturally the two methods present big differences as the hip and the knee have very different geometries and the number and position of the strain gauges is different. Nevertheless, it explains its nature.

The software solution for the hip project, described in 1988 [BER88], was designed according with natural constraints from that time. The requirement for a "fast program" at that time does not have the same meaning as it has today. The routines were then written in assembly to "read the signal times, identify the temperature channels, synchronize the signals, and search and correct possible transmission errors by checking signal limits". All this processing required fast software for the time the project was done. Fortunately there is today available better processing power and higher range of software development environments to work with. The software solutions from the hip project could not be considered for this traineeship.

Important to consider, though, was the already existing technologies that were used along this project. As it was described in figure 10 (Problem Description section) the front-end application to control the calibration rig was made in LabVIEW. The version nr 5.0 was used. At the moment this traineeship started, LabVIEW 6i was available to use for the project. It should be referenced that the version 7 was commercially released during the traineeship and it is a significant improvement in the software. For an instance, the techniques available to construct the interface have changed dramatically. The version 6i did not allow control of user events programmatically, taking out some exceptions. There was not implemented a structure to detect, per example, the coordinates of mouse click in a defined surface. To know if the user clicked in a determined button it was necessary to read the button state (on/off) in time intervals, a technique known as "polling" that relatively wastes some CPU resources. The use of the version 7.0 could make a big difference in the development of the software for the project and it was necessary to analyze if it was worthwhile to purchase the new version.

3.1.2 Communication System

The developments of the application “Mk2” and “Mk4” (see figure 10 – Problem Description section) was done in Microsoft Visual C++ 5.0. These programs act as server applications that read data from the serial ports and give it to other applications through the Dynamic Data Exchange (DDE) Link. In this case only a LabVIEW application acts as a client but any application that uses the DDE link and uses the correct protocol can request information. In that way the application to control the calibration rig can run concurrently with the servers “Mk2” and “Mk4”. With this approach, according with one of the programming authors [HER00], the client application gets the incoming data only when is needed without stopping the execution of the principal application.

The DDE technology is an interprocess communication system that is especially
suitable for applications that use real time data. It occurs always in a client-server paradigm and the client can request data at a particular time, or request to be notified when there is changes in particular set of data, or receive updated data after changes.

Given its characteristics, the developers of this application chose it for a communication that occurs asynchronously using then the most of the available resources.

Although it is now an outdated technology, it is also tested and known enough to avoid most of the problems. During this traineeship there were no reports of faults in the data acquisition by the server applications.

In this analysis it should also be referenced that the use of the Visual C++ 5.0 and the API of Windows for DDE will constrain any change in the operative system if it is decided to reuse these applications.

3.1.3 Algorithm Development For Definition of Forces In The Knee Joint

To study algorithms and test new ideas about the development of algorithms that would transform the strain data in Forces and its position of application in the implant Microsoft Excel seemed to be good platform to work with.

The mentioned data was the result of a Finite Element Analysis (FEA) applied to the tibia prosthesis model. Another member of the research team was developing these algorithms using data from the calibration rig that produces a bigger set of data from a real implant. Nevertheless, the use of the FEA data could give a starting point to test ideas since its results are not dependant of the implant construction faults and mechanical problems. Once that it was pure mathematical data it was easier to work with in a sense that didn’t require further tests or analysis to verify its correctness. Some statistical analysis was done before with the calibration rig data and use it with the FEA data could be a good option.

The technologies to study these algorithms also depend naturally of the ideas behind it. So if at the beginning for mere statistical analysis a spreadsheet tool like Microsoft Excel would do, for a solution based in neural networks, another tool would be necessary. A report from a previous student mentions this idea [HER2000]. He too made some analysis on the data from the calibration rig. Those analyses became outdated due to modifications in the design implant during a previous phase of the project. He used a software package called NNFit 2.0 to study about neural networks and to make decisions if neural networks could or not be used. NNFit 2.0 was used, between other reasons, because a certain convenience in the input data format that had similarities with the calibration rig output data format. As he refers, there was already available for use by this department the Neural Networks toolbox for Matlab 4.2. Any of these two programs could be used to test algorithms based in neural networks during this traineeship.

3.2 Technological Options

The definition in the beginning of the technology to use for this traineeship would have a major impact in the model and the methods to use in the development of the software.
It was not only a problem of defining the computer language, two completely different development paradigms had to be compared and evaluated for this particular project.

The main application to be created, the user interface, had special requirements and Visual Basic, C++ and LabVIEW were considered to its development. It was already defined that Microsoft Windows would be used as the Operative System due to the project history for an easier integration. As it was used before, C++ could be used through the Microsoft Visual C++ development environment. For each of the basic requirements the suitability of the alternatives was analyzed (the original form of the requirements can be seen in appendix A). All comparisons that follow with LabVIEW are made considering the available version during this traineeship that is the 6i version.

1 - The later integration of algorithms for data interpretation that were to be developed by other member of the research team.

As it was probable that external code would come using C++, making the integration also in C++ would make it easier and it could be done faster. LabVIEW supports the integration of C code through what is called the Code Integration Node (CIN) that is a node with functions written in C that are compiled with the G language code. For the LabVIEW case it was still possible to use the C++ code for the algorithms since it also supports calls to external libraries, in this case a DLL could be used. Visual Basic would encounter itself in a similar situation. Any interprocess communication supported by Microsoft Windows could be used.

2 - Visual system to record and show the received data

In this case the most appropriate would be one of the textual languages. This requirement demands a management of data with similar properties to a database management system but not so complex. Data about forces and the coordinates of the point of application will be available every 10 milliseconds forming a large file of data per each session. Each session by itself will correspond to patient and an activity carried by him. Other data about the patient must also be available. All this data must be organized and available to be visualized by the application by choice of the user. An implicit requirement comes when entering this data is necessary. Because modifications on some sensible data about patient details should not be allowed to every user, a super user profile must be implemented in the system. Only the super user can make changes on patient personal data.

LabVIEW does not give support for high level of security. Implementing the super user functionality in LabVIEW is hard and will never be as efficient as the other languages here.
3 - Graph plots to view the results of forces and coordinates of application point in real time and from saved data.

In this case LabVIEW presented a huge advantage relatively to the textual languages. Because it is a development environment prepared for software applications to laboratories and research it presents a range of built in components for graphical plotting. It also gives interesting functionalities to use with the graphical plots (per example, easy to use zoom in/zoom out). Developing these components and some of these functionalities in the considered textual languages would be cumbersome.

The analyses made for this last requirement is a heavy weight argument in the discussion about the technologies to use.

The fact that LabVIEW has been used extensively during this project means also that other research team members will not have major problems in using the application or even analyzing parts of its code. It also simplifies integration with other applications of the project and there is a good possibility of code reuse.

LabVIEW was the chosen platform to work with although there are risks that should be considered and are discussed in the next section.

3.3 Technological Risks

"It is easy to create an attractive GUI in LabVIEW, but something of a skill to create a responsive one." [GRA03]

LabVIEW presents its greater risk in the lack of interactivity and flexibility to create custom-made user interfaces. It provides the required software components for this project but it allows only a certain degree of customization that should be studied.

Another problem can be inherited from its programming paradigm. It is known that showing so much information in the screen it is not always a good idea. The "Deutsch limit" defines:

"The problem with visual programming is that you can't have more than 50 visual primitives on the screen at the same time."

This happens because there is a readability problem with so many icons and wires passing in the screen. This is graciously called the "spaghetti" code.

Therefore, structuring the code can be an important issue and it has to be analysed what does LabVIEW provide to help in this way.

These factors will influence the approach used to build this application. To try to eliminate or diminish some of the risks, tests or prototypes will be made trying at the same time fulfill some of the requirements. In this process the validation of the prototype by the responsible of the project will be important.

LabVIEW is based in primitives that are the building blocks of a Virtual Instrument (VI). These VI's follow a hierarchical structure that defines the structure of the code.
However this is the general use of LabVIEW. There is no flexibility in the structure that allows better adaptation for different problems. This point is better discussed during the next three chapters.
4 Development Model of the Front End Application

To understand the development of the applications is first necessary to consider the followed model. The model is thought taking in considerations the chosen technology and its risks.

The choice of LabVIEW seemed the most appropriate as it was described in the previous chapter. But for a smoother development of the front-end application it was fundamental to identify the dangers that one must face in this choice. It must be noticed that LabVIEW, as a pure visual system, has its own programming paradigm.

In a generic way, pure visual systems are known to have inherit the following problems:

- Lack of flexibility – per example, in building a responsive user interface and create data structures.
- Lack of extensibility – Every other VI is related with a superior on until the main VI. Any changes in the main VI can cause changes in other VI’s and vice-versa.

In order to eliminate the technological risks described and control some of general problems of visual programming a model of development was created similar to a software spiral model of development but in more simplified version. Mainly three prototypes were created. Each satisfying a set of requirements and basing itself on the previous. The final prototype should satisfy all requirements completely and more than that satisfy the client to whom the application is developed.

It should be noticed that the specifications of the applications are based in Data Flow models that is natural in visual programming instead of the Control Flow models of the conventional textual languages. More specifically the used design specifications model for Lab VIEW is well documented in [BRI01]

This model would also allow the client, to whom the software is developed, in this case the responsible for the entire project, to verify its evolution and to change requirements along the development after seeing some results. This is even more important since that the application to be developed has as major component the visual interface.

The general-core of LabVIEW application is the state machine. This also justifies this model since in a state machine there is a range of cases whose condition is tested. This test will tell if the case is executed or not. Add functionality to the application is also adding extra test cases. This should avoid having to change other VI’s below corresponding to the execution of other cases.
In figure 11 is shown an example of a state machine. The outer rectangles are the correspondant to “while” cycles. In this case they execute always while the application is running testing the conditions defined by the inner rectangles (the picture shows only the default case).

In the second picture it is added a state transition. The number 1 in the bottom picture defines the next state to text.

This means that adding functionalities implies adding also more cases and VI’s to be executed under each case.

In conclusion the general way of building applications in LabVIEW also approves the chosen development model.
5 System Architecture

It is given in this chapter a general vision of the conceptual architecture and development process.

During the development process of the main application, a range of prototypes was built until the requirements were satisfied. In this chapter the specification of these prototypes and the requirements that they were intended for are described.

It was the first prototype that validated the choice of LabVIEW to develop this application. It is a simply prototype that implemented the main functionality: opening of saved data for plotting in the interface.

The latest prototype was designed to cover conceptually all the functionalities.

5.1 Architecture principles

Considering the choice of LabVIEW to develop this application and that it was a completely different programming paradigm it was necessary to review the architecture of the application. Mainly how is it possible to keep flexibility, extensibility, maintainability, code reuse, and readability in LabVIEW?

As it is known that future components will be added later (algorithms to convert strain data in applied forces and position of application) flexibility and extensibility have a major role. To increase the importance of this issue, another point to consider, is that with the evolution of the research project, changes in requirements are inevitable.

The structure of the code should also allow maintainability once that not only other functionalities would be added in later phases of the development but alterations to the implemented functionalities will also be made.

Since there is the intention of providing a complete application it is not expected that code reuse is an important issue nevertheless it should be considered for a faster development.

A good conceptual software structure should provide levels of abstraction and, by that way, improve readability.

An application in LabVIEW should be divided in three tiers [BIT01]. The first tier is referred to as the “Main Level”. The Main Level consists of the user interface and the test executive. The second level is the “Test Level” or the “Logical Level”. The Test Level is responsible for performing any logical and decision-making activities. The lowest level is referred to as the “Driver Level”. The Driver Level performs all communications to instruments, devices under test, and to other applications.
5.2 First Prototype

With the first prototype it was intended to present the crucial components right in the beginning. From each prototype it would be possible to extend the components and add the functionalities for the rest of the requirements.

The functionalities to test with the first prototype were:

- Open saved files with formatted data with forces magnitudes and coordinates of its application point.
- Close files with the formatted data.
- Guide the data to the respective graphs to plot the data according to a specific timing.

From the most interesting capabilities of these components, the built in graphs would allow to

- Define the color and style of the plot(s). Resize the legend to display multiple plots.
- Define labels for scales and configures scale properties
- Scroll through the data in the graph.
- Zoom in and out on the data in the graph along the X and/or Y axis.

The user can customize these options that are already built in the components.

The prototype was implemented successfully and, despite its simplicity, it is possible to take the first conclusions.

The prototype has shown that LabVIEW is flexible enough to satisfy the main functionality that is the data plotting for user analysis. It also adds interesting extra functionality that can be useful for the user.

5.3 Second Prototype

The evolution from the first prototype to the second encountered some structural problems. This prototype was mainly designed to complete the graphical requirements and make its relation with the patient data. It is still concentrated in the visual interface requirements and should allow a certain degree of interaction.

From the patient data, the requirements to be satisfied would be:

- Specific Control for the list of patients – This object in the interface should have to open a list of patients and allow its selection.
- Creation of the new patient – Object in the interface to open the saved list of patients.
- Activities Table – This object in the interface allows the visualization of data related to the patient. In this table each row will correspond to a defined activity and each column to a date. The intersection of rows and columns defines a session. It should be possible to edit this table and allow interactivity to the user. The user should be able to select a wanted session from this table that would be automatically plotted in the graphs. This later requirement was defined after the first prototype.

From the Implant data, it should be possible to:

- Visualize the data in different graphs for each compartment of the implant. The graphs to be displayed were the data for Axial Force, Axial Torque (later eliminated), Antero- posterior Shear Force, Medio-Lateral Shear Force and the combination of Axial Force and Shear Force. In one special graph it should be allowed the direct visualization of the point of application of the forces plotted in the picture of the implant. This special graph would have to record history of
the latest three past coordinates data and plot it in different color (forming a
grad-net). That way, with the plot of data from four different time intervals, it
was more clear the orientation that the point of application of the force was
following with the time. This special requirement was defined after the first
prototype.

- Control the speed of the data plotting. In real time each graph will have to be
  actualized with new data each 10 milliseconds. The user should be able to
  modify speed for a better analysis of the data.

- Controls to clear and stopping the plotting at any moment.

This list of requirements is added to the already satisfied ones in the first prototype to
form the second prototype.

However during the implementation of some of the requirements described here it
became clear that a better specification was needed.

Extending the previous prototype for new functionalities became cumbersome. Errors
became hard to solve as the readability of the code was decaying.

As by itself, implementing necessary interactivity was already difficult due to the
visual programming paradigm (see previous chapters), it would became much harder if there
was structural errors. To analyze this issue UML diagrams for use cases were built.
With a discussion of the requirements with the client, it became clear that the system have to be designed for two types of user. The requirements were grouped in this specification and a more clear structure of the data became visible. Although the patient data has to include the implant data (each implant belongs to a patient). There is a group of requirements only connected to the patient personal data and other connected specifically to the implant data. The patient table data makes the relation between the two types of data. This table has a list of sessions that correspond to an activity carried on by the patient in a certain date. Each session produces the implant data that can be plotted – “plot patient knee data” use case. The user can update this table with the “manage patient table data” use case.

The “retrieve implant data” use case expresses the functionality of opening the data
that is connected to the identification of each implant: owner and on which leg inserted – left or right.

There must be in consideration, that later in the application external code will be inserted to get data sent by the implant inserted in the patient. This code will be responsible to acquire the data from the implant and transform it to the data used by this application. This corresponds to the operations carried by algorithms that will determine the forces and its point of application in the implant from the strain gauge data. Therefore the diagram includes a second system (Implant System) that is the source of this data.

Still according to the design phase, a class diagram was built.

Besides it helps to clarify the required data and its relationship there was also the possibility of using a new toolkit for LabVIEW. The toolkit called GOOP tries to improve the structure of a LabVIEW application using ideas of the Object Oriented programming paradigm. That option was discarded as it implied too many changes at the time.

This Class diagram provides only the attributes for the objects and relationships once it is only a design diagram. Methods would be added later for a development phase. However this was enough to give a new direction for the third prototype without require more specifications.

![Class diagram for a new prototype](image)

Figure 14 – Class diagram for a new prototype
The **patient class** is core of the diagram. It has as the attributes the patient personal data. Each patient will have a table of activities as defined by the classes **table** and **activities**. Each activity has **parameters** to define it. Other information modeled in the diagram is the data related with the **knee implant** for which each activity will cause a **measurement**.

### 5.4 Third Prototype

The last prototype would encounter itself already with the structure of what will be the final application.

---

**Figure 15 – Module Specification in the three Levels of the chosen design hierarchy. The modules are called from up to bottom but data flows in both directions.**
2- Plotting speed control

This control should modify the speed of the displaying data on the “Graph plotting” controls. It should allow the real time speed, plotting of new data each 10 milliseconds and allow slower speeds.

3- Acquired data control

This control should contact the Logic Level to retrieve the required data of the implant. This data must be formatted to be visualized in the “Graph plotting” controls.

Logic Level

This level is called by the main level and the modules should be used to verify the availability of the requested data and provide it if it exists. These modules call the modules in the Driver Level if necessary.

Delivering of saved implant data for visualization – This module is called to deliver data to controls that show the implant data: forces and its position of application. Because the data is to be shown in different type of graphs, it has to be formatted according the one that serves. It calls the driver Level for retrieving the stored data in the database.

Opening of patient data – This module allows the visualization of the patient personal data. This data comprehends: Name, Number, Birth date and a table of activities that is linked to data file for each session of each activity. It calls the drivers for storage.

Update of patient data – This module uses the same data of the previous module with the intention of saving it. It calls the drivers for storage.

Driver Level

Drivers used for file I/O and error verification.

5.5 Conclusion

The latest prototype finds itself conceptually near the design of the final application once the modules cover all the requirements.
6 Development Description

The conceptual specification of the prototypes and the problems of development were indicated in the previous chapter.

This chapter intends to show the physical model of the developed prototypes, call attention to the errors previously indicated that are immediately visible and show how the situation was greatly improved in the final prototype.

The functionalities implemented in the user interface are also here presented.

6.1 Prototype Development

LabVIEW uses primitive objects as the building blocks for an application. These building blocks are the same as the primitive functions offered by conventional textual languages.

To provide structural capabilities, LabVIEW uses the concept of Virtual Instruments (VI’s). Each VI is built with primitives and/or other VI’s. In the end, an application should have a well defined hierarchy of VI’s as defined previously in Architectural Principles.

The first Prototype presented only File I/O and graph plotting with almost no programming for interactivity. It served his test purpose.

The second prototype presents a configuration of VI’s shown in figure nr. 17. In the top of the hierarchy is the main VI called in this case “UI 3”. Each VI has associated a control panel that provides the data input/output for that VI. If it is the main VI, that control panel is immediately visible to the user forming its user interface with which he can interact. The main VI, as a normal VI, calls the lower VI’s to perform operations and provide him the output.

In this prototype, all VI’s below “UI 3” are VI’s to provide Input/Output or test for errors in the same operation. Some are directly driver VI’s (the ones that don’t have the blue arrow in the upper left corner) others call only a set of Drivers to perform the operation in a sequential manner. This means that all the logical operations (from the Logical Level or Test Level) are inserted in the main VI. As it was explained in the previous chapter along with conceptual analysis of this prototype, this constitutes a problem

In the next section it is possible to see how this was improved.
6.2 Description of the Final Prototype

This prototype was achieved with the improvement of the second prototype and the help of validated clarifying documentation. In this section the prototype is described and compared with its conceptual view.
6.2.1 Structure Overview

The final prototype has improved in its structure according to figure 18. In figure 17 is possible to compare its conceptual structure.

![Diagram showing structure overview]

Figure 17 - Relationship between the ideal architecture and the physical structure

At the main level, this prototype includes mainly a tab control where the already identified data groups are divided and it is assisted by a menu system. It allows the user to manage the data he is actually interested in, providing a better organization of the interface.

At the Logical Level, it still includes some of the problems of the previous prototype, but its organization was greatly improved pulling to the lower VI's the File I/O responsibility.

6.2.2 Physical Description

The actual physical model of the VI's corresponding to the Logical Level is shown in figure 18.

![Diagram showing physical model]

Figure 18 - The Physical Model corresponding to the Logical Level of the final prototype

It is possible to see that some of VI's that should be in the Driver Level are still
organized in Logical Level. These are all the VI’s in complete white, starting in the number eight (from left to right). These VI’s were implemented in the previous prototype.

The VI’s implemented in this prototype are the first seven, identified by a little chart in the picture, described, respectively, as follows:

2. DPathParameters.vi – responsible for record configuration parameters
3. FAdminLogin.vi – responsible for recognizing the Admin (Super User).
4. FOpenPath.vi – responsible for access File I/O Drivers for data income.
5. FTableModify.vi – responsible for saving data from the Activities Table
6. line generator.vi – responsible for formatting data for graphs in the Graphical View
7. NPatientDia.vi – responsible for saving a new patient data.

The next picture shows the complete model with all the drivers.

![Diagram of the complete physical model](image)

**Figure 19 – UI 5 – The complete physical model**

### 6.2.3 Visual Interface

Each VI has a front panel were the user can interact with VI. Every component inserted in the code of the VI has a visual part in the front panel. The interface of this
application is constituted by the front panel of the main VI, front panel of called VI’s (VI’s whose name start by ‘D’) and a menu system inserted programmatically in the LabVIEW platform.

From the components of the interface, only the tab control does not require the “polling” technique mentioned in chapter 3. All buttons and menu items are programmatically read with a time interval. If there is a change in their state, the correspondent case in the state machine is executed. One of the problems of this technique is defining the time interval in practice. Using a time interval too short and it is possible that it doesn’t recognize that there was change of the state at all. Using one too long and it may happen that the state machine is missing interactions. The correct timing for polling is defined with tests in the programmer’s computer.

The front panel of the main VI includes tab control that divides the accessible information according the analyses previously presented: Implant Data, Patient Data and Graphic View.

The menu system (figure 20 – the last two items - Application and Options were created; the other part of the menu belongs to LabVIEW) provides secondary options as path parameters and administrator log in/log off.

6.2.3.1 Patient Data

![Figure 20 – Visual Interface – Implant Data](image)
In figure 20 is possible to see the tab control (just below the menu) with the applied organization. The patients are automatically listed once the application is opened in the combo box. That list is read from a file according to path parameters introduced.

The main component of this part of the interface is the activity table that can be open for each patient using the button “Open” on the top. This table only shows the sessions done for the patient but only the administrator can modify it and save.

Each session corresponds to an activity/date combination (rows/columns). For each session there is a file that associated that contains the data measured in that session.

To Login as administrator is possible to use the “Application” item of the menu and choose “Admin Log in”.

![Figure 21 – Menu System](image)

Figure 21 shows the menu used in LabVIEW applications and the last two items inserted programmatically. Here the administrator (super user in the specification) can Log in and access new functionalities.

![Figure 22 – New functions for the Admin (Super user)](image)
Now that the administrator is logged in he can access to the patient data and change the values in the table and save it.

As according with the requirements, here it is possible to scroll through the patient list and:

Create New Patient - on the bottom, with the button “New”.
Remove a Patient – with the button “Remove”.

The list is only saved after pressing “Save Patients”

On the top of the screen the new two buttons “Modify Table” and “Save” allow the respective functionalities with the activities table.

As explained, each session has a file associated that contains its measurements. To lessen the difficulty of search that file for each session and plot it, interaction was added to the table. It is possible to open any file from the table with a mouse click.

![Figure 23 – Opening a file with mouse click in a table cell corresponding to a session](image)

The system asks for the confirmation of the user. It is possible to see a little triangle in the top left corner of the cell in the first column (“70”) and forth row (“A-Resting”). This
triangle indicates the position of the cursor in the table and, in this case, where was the click of the user.

LabVIEW is not ready for this type of interaction. It does not recognize the position of the mouse clicks in surfaces. This functionality was added playing with the initial and final position of the table cursor. A “polling” is made to this cursor in the state machine of the main VI. From here one can deduce, that “polling” can be an art by its own.

The file will be opened for graph plotting.

6.2.3.2 Implant Data

Once a file is opened for plotting the tab control is changed for the implant data. Changing the display to another tab page is not allowed programmatically. This illusion is created making the other tab pages invalid. Once there is only one page to show, LabVIEW displays it. The other pages became valid right after without the user notice what happened. This lack of flexibility in LabVIEW confirms the first sentence of the section “Technology Risks” in chapter 3, referring how difficult it is to add the required interaction in the interfaces.

Figure 24 – Implant Data – View of the user interface
Figure 24 presents the view where the user can access to the implant data. Here is presented two charts for each compartment (left/right).

Each graph is plotted against time and has two Y-axis. This allows a better visualization of multiple plots. For the top graphs the left axis plots the Axial force which magnitude is normally higher than the shear forces plotted according to the right axis. The bottom graphs show the relation between the resultant force (left axis) applied in the implant and the point of application (right axis).

These graphs are components integrated in LabVIEW that allows a good degree of customization for this particular case fulfilling the necessary requirements for plotting the implant data as expressed in chapter two under goals and requirements.

The extra functionalities offered by these graphs are interesting for the user making these components one of the bigger strengths of LabVIEW for this application. Just to give some examples, with these graphs is possible to zoom the plotting in any direction by clicking in the magnifying glass below them. It is possible to scroll easily along the data with the mouse or with a scroll bar. Digital values of the plotting are also presented at the same time.

These, between others, functionalities give to the user a more capable interface for data analyses.

Above the graphs are include buttons to stop the plotting, clear the graphs and control their speed of plotting.

At the moment the speed of plotting is not fast enough for plotting in real time. In the right corner of the screen there is one variable, called “Delay”, being shown temporarily. This variable identifies the time, in milliseconds, required to add new data to the graph.

The maximum delay, in real time, should be 10 milliseconds. This speed depends greatly of the hardware capabilities of the computer running the application. However it was show that a computer with a CPU of 1.7 GHz and 512 MB of main memory the delay would go to 16-18 milliseconds for this plotting. However only 70% of the processing power was being required and the memory used was much below the maximum indicating that the bottleneck was on the graphical hardware.

6.2.3.3 Graphic View

For a more direct visualization of the results obtained with the measurements of the strain gauges. A more advance view is given in the third page of the tab control as defined in the specification of the previous chapter. The next picture shows this page where the forces and the point of application are plotted directly in pictures of top part of the implant.
Figure 25 – User Interface – Advanced Graphical View

On the top part are present three views of the implant showing its compartments and a red line (in diagonal) that indicates the magnitude, direction and the point of application of the resultant force being applied in the implant at the time. A blue line is inserted and defines the axial force. There is also a green line (not visible in the picture) that identifies the shear force.

At the bottom a scheme of the superior view of the implant is presented. An orange dot, inside the lines defining the compartment, indicates the exact point of application of the force.

It is possible in this screen to visualize the delay of the plotting of new data. In this case the values vary between 600 and 700 milliseconds. This also indicates that the problem is in the graphical display of LabVIEW applications and not in the software developed itself once the previous tab page presented plotting graphically less “heavy” at faster speed.

Again this situation was tested with previous referred computer decreasing the delay for 80 to 90 milliseconds.

Further tests will be made indicating if improving the hardware responsible for graphical display also decreases the delay.

For visual interface purposes, LabVIEW seems here to allow a very expressive visualization. It also should be referred as strength of LabVIEW for this particular case.
6.3 Conclusion

The program faces at this moment a speed problem that it seems to come of the poor ability of LabVIEW to display its visual interface. To verify it a better graphic board will be added to a computer and the speed will be observed.

Computers with better processing power and main memory were used previously. The results were improved but not the sufficient. However the resources of the machine were too way below the maximum in these two issues. This indicates that improving graphical performance should solve the problem.

Taking this problem aside, the software fulfills the functional requirements and gives the user extra tools for his analyses.

It is also a very expressive interface once more showing that LabVIEW is a capable tool to create good looking software.

The expected problems in extensibility were solved with better specification, using UML diagrams, and using a more proper structure for LabVIEW applications.

Another predicted problem was the flexibility of LabVIEW that gave some work to overwhelm. Adding interaction to its attractive interface can sometimes be difficult. This was clearly shown when there was a try to control the display of the pages of the tab control programmatically given some actions of the user. This situation was solved by making all the other pages inactive, forcing the system to display the only page active at the moment, and then making the rest of the pages active again.

It expected that the Version 7 of LabVIEW has improved the flexibility of the platform. Once it recognizes user events adding more interaction to the interface should be much easier.

Nevertheless the problems in adding interaction capabilities, the end result of the visual interface, the functionalities it provides and its expression abilities show that LabVIEW was the correct choice to develop this application.
7 Modification of the Communication System

The communication system is the part of the research project that supports receiving and sending data to and from the implant inserted in the patient. The computer receives this data through the serial port.

The communication system intended is in everything similar to the one used in the calibration rig. In this case the program “Mk 4”, written in C++, is responsible for synchronizing with the external data source, receiving the data and decoding it. After decoding it passes to the LabVIEW application, through the DDE link, as a string.

Because there were changes, with the development of the research project, in the data format that the computer receives, changes in the software that take care of the communication had to be made.

After some study of the programs and some changes made to visualize the new information it was discovered that the source (patient box) was not sending the data in a correct manner. The changes were put on stand by until the person responsible by that part of the project fixes the problem.

7.1 Data Decoding

The frames received by the “Mk 4” program have the following structure:

<table>
<thead>
<tr>
<th>Frame</th>
<th>AuxBits 16-13</th>
<th>AuxBit 12-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000 sync</td>
<td>Right Overshoot</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>Most 2\textsuperscript{nd} Left Temp</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>Most 3\textsuperscript{rd} Left Temp</td>
</tr>
<tr>
<td>3</td>
<td>0100</td>
<td>Most 4\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>4</td>
<td>0110</td>
<td>Most 5\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>5</td>
<td>0111</td>
<td>Most 6\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>6</td>
<td>0000</td>
<td>Most 7\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>7</td>
<td>0001</td>
<td>Most 8\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>8</td>
<td>0010</td>
<td>Most 9\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>9</td>
<td>0100</td>
<td>Most 10\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>10</td>
<td>0110</td>
<td>Most 11\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>11</td>
<td>0111</td>
<td>Most 12\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>12</td>
<td>0011</td>
<td>Most 13\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>13</td>
<td>0000</td>
<td>Most 14\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>14</td>
<td>0011</td>
<td>Most 15\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>15</td>
<td>0011</td>
<td>Least Overshoot</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frame</th>
<th>AuxBits 16-13</th>
<th>AuxBit 12-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000 sync</td>
<td>Right Overshoot</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>Most 2\textsuperscript{nd} Left Temp</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>Most 3\textsuperscript{rd} Left Temp</td>
</tr>
<tr>
<td>3</td>
<td>0100</td>
<td>Most 4\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>4</td>
<td>0110</td>
<td>Most 5\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>5</td>
<td>0111</td>
<td>Most 6\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>6</td>
<td>0000</td>
<td>Most 7\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>7</td>
<td>0001</td>
<td>Most 8\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>8</td>
<td>0010</td>
<td>Most 9\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>9</td>
<td>0100</td>
<td>Most 10\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>10</td>
<td>0110</td>
<td>Most 11\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>11</td>
<td>0111</td>
<td>Most 12\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>12</td>
<td>0011</td>
<td>Most 13\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>13</td>
<td>0000</td>
<td>Most 14\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>14</td>
<td>0011</td>
<td>Most 15\textsuperscript{th} Left Temp</td>
</tr>
<tr>
<td>15</td>
<td>0011</td>
<td>Least Overshoot</td>
</tr>
</tbody>
</table>

Figure 26 – Table showing the auxiliary bits sent per each frame

It receives 38 bytes in each frame that include information for 24 strain gauges values (with 12 bits allocated for each), synchronization (4 bits) and auxiliary data (12 bits).

Each word of 8 bits is received with a start bit and a stop bit. In total 380 (38 bytes * 10 bits/byte) plus 4 extra bits flow through the communication channel for each frame and are received in the serial port. Using a baud rate of 38 400, each frame is receive in 10 milliseconds.
A total of 16 frames form a major frame as seen in the previous diagram. The information being sent was modified in the seventh frame of the major frame where there were 4 bits not used and a “personality” byte. These 12 bits correspond to the auxiliary data sent by the seventh frame of the major frame.

The information to be received now in these auxiliary 12 bits is:

- The patient number, and left/right leg bits that are packed over the 4 current spare bits of frame 7
- The 2 voltages and the event indicator are packed into the 8 current ‘personality’ bits of frame 7.

The protocol for these bits is as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>identifies start of voltage data in bits 1,2</td>
</tr>
<tr>
<td>1</td>
<td>lithium ion battery voltage</td>
</tr>
<tr>
<td>2</td>
<td>SMPSU voltage</td>
</tr>
<tr>
<td>3</td>
<td>external event marker</td>
</tr>
<tr>
<td>4</td>
<td>bit to synchronise external event (indicates the frame in which the event occurred)</td>
</tr>
<tr>
<td>5</td>
<td>bit to synchronise external event (indicates the frame in which the event occurred)</td>
</tr>
<tr>
<td>6</td>
<td>bit to synchronise external event (indicates the frame in which the event occurred)</td>
</tr>
<tr>
<td>7</td>
<td>bit to synchronise external event (indicates the frame in which the event occurred)</td>
</tr>
</tbody>
</table>

The “Mk 4” program, after an initialization process for synchronization, receives each frame until it gets a major frame (16 normal frames). Once it has a major frame, if there is a client connected (in this case the LabVIEW application), sends it as a string.

Because of the limited number of bits available for all auxiliary data, the additional data from the patient box is being encoded serially over several major frames. This is possible because it can be updated more slowly that the strains.

Because it is the “Mk 4” program that interprets the received data, LabVIEW does not have in this string the information of the previously unused 4 bits of the auxiliary data of 7th frame. However it is possible for the LabVIEW application to interpret the personality bytes according to the new format without any changes in the Mk 4 program.

Next section presents the modifications in the LabVIEW client.

### 7.2 Software Modification

A LabVIEW application that receives the Mk4 data and displays it in ‘real time’ is called “Mk4cl_plot” and this routine was modified to allow the visualization of every bit of the personality byte with the new information.

The physical model of the application is shown in the next figure.
The Personality VI was inserted and receives the auxiliary data from each frame. In this VI the characters are again translated to bits. To allow the visualization of every bit received in certain time interval, the data is recorded in a text file through VI’s below to the personality VI.

After analyzing this data in the text file it was shown that the data incoming through the serial port was not the intended data sent by the external source.

To analyze this situation and to analyze if the "Mk 4" program was the source of the problem, a LabVIEW application to read the serial port was developed. Its VI’s are shown in the next picture.
Figure 28 – Physical model of the serial port reader

This VI only uses drivers to read a serial port and to control file I/O. It is a VI to be used in the Logical Level of an application.

After testing it was shown that the data received by the simulated patient box was not being sent correctly and that there was not fault in the LabVIEW application or the Mk4 program. This situation is still unresolved, awaiting the outcome of further work.

7.3 Conclusion

In this case the problem was well identified after study of the applications that required a reasonable amount of time in the traineeship.

This modification is in stand by until the person responsible for the creation of the simulated patient box can correct the problem. This is expected shortly. After that the changes in the LabVIEW application will continue.

Changes will be made to the “Mk 4” program to pass its string the previous unused 4 bits of the 7th frame of the major frame.
8 Algorithm Development Based in FEA

Finite Element Analysis is a well-known and widely used method for the numerical study of stresses in a structure by considering it to be comprised of discrete elements. This study was applied to the implant model and results were taken for a range of applied loads. The intention was to simulate the forces applied by the femur in the implant and to know the resultant stresses at the positions where the strain gauges are found. In that way it was possible to have a theoretical perspective of the strains caused by those loads and to develop ways of combining those strains to yield the forces.

Using a theoretical, though small, set of the produced strain it is possible to have a good insight to the behavior of strain gauges when loads are applied.

First the formulas to calculate strains, used in previous phases of the project, were analyzed and modified. The result was a better approximation to the values when a real load was applied to a real implant. More details in “Calculation of strains” section.

A matrix of sensitivities and cross-sensitivities was then derived from the resultant information of a pattern of applied loads, an idea that also came from previous work. With this matrix, using a reference point, it was possible to have an approximate distance to the real point and the applied forces. It was shown that using an iterative process, and modifying the reference point, that distance would eventually iterate to zero, which would return the magnitude of the forces, axial and shear and the position their point of application. Tests show promising results. This is presented in the section “Matrix Method”

Unfortunately, during the traineeship it was not possible to get concrete results to finally evaluate if this method can be applied or not in practice. However, at the time this report is being written, more tests with a larger set of data are being carried out.
8.1 Calculation of Strains

Given the dimensions and design of the prosthesis, a set of load cases was applied to strategic points in the implant model. It returned the resultant stresses at the 24 points of the strain gauges according to a local axis system.

Figure 30 – Axis system for each half bridge of strain gauges

From the stresses at these points and considering its axis, it is possible to know the produced strains at the same point and for another axis system. In fact, due to modifications in the implant the considered axis for the strain gauges in the real implant and the model used for FEA is not the same.

Because the strain gauges only measure strain according to the direction they are collocated (consult appendix C) it was important to calculate the stresses for the axis defined for the real implant.

Each stress value reported by the FEA for each one of the 24 strain gauges was transformed to return the stress according to the defined axis in the implant for the target strain gauge (more information about how this is done can be seen in appendix B). The next pictures show the formulas used.
Figure 31 – Formulas for calculating new stresses when there is theta rotation

\[
\sigma_{nn} = \sigma_{xx} \cos^2 \theta + \sigma_{yy} \sin^2 \theta + 2 \tau_{xy} \sin \theta \cos \theta
\]

\[
\tau_{nt} = -\sigma_{xx} \cos \theta \sin \theta + \sigma_{yy} \sin \theta \cos \theta + \tau_{xy} (\cos^2 \theta - \sin^2 \theta)
\]

The first picture indicates the stresses applied in a cube and gives the axis system for the stresses in an inclined plane rotated theta degrees from the original system. On the right the stresses to calculate are shown:

\(\sigma_{nn}, \tau_{nt}\)

The bottom pictures show how the formulas are obtained.

Once we have the stresses for the correct axis system, the strain is calculated along the X-axis. This is the only axis from which the strain gauge can measure the strain. Previous work has used the following formula,

\[
\text{Strain in X-axis} = \frac{(X_{\text{stres}} - Y \times Y_{\text{stress}})}{E}
\]

Where \(V\) is Poisson ratio and \(E\) is Young’s Modulus.

The formula used only X and Y stresses (the Y stress causes the Poisson effect along X axis) because the strains gauges are inserted in a surface of the implant. This is called to
assume the plane stress condition. In the circumstances of plane stress condition there is no Z stress since the structure where the strain gauges are inserted is considered a plane and therefore only has two dimensions.

However, in this particular case, the FEA shows that it is not absolutely correct to assume a null stress along the Z-axis. Some loads seem to produce a relevant amount of stress at some points of the strain gauges location. Despite the fact in some cases this stress is very near zero, the formula was changed to include the Poisson effect caused by the third dimension. The formula became.

\[
\text{Strain in X-axis} = \frac{(X_{\text{stress}} - V*Y_{\text{stress}} - V*Z_{\text{stress}})}{E}
\]

Where \( V \) is Poisson ration and \( E \) the Young’s Modulues

The next figure shows a practical case of visible improvement,

![Graph showing strain comparison](image)

**Figure 32 – Comparison between results of new formula, old formula, and calibration rig data**

The line with triangles represents the results of the new formula with the new dimension. The old formula is represented by the line with squares and the calibration rig data by the line with little balls.

It is possible to see in this figure that there is a slightly better approximation to reality with the new formula. There is likely to be a difference between strain values found from an applied load in the FE model and those measured in an implant with the calibration rig. This is caused by several factors that can produce a deviation, such as: differences in the angle of orientation of the strain gauges compared with those defined in the model, differences in the geometry of the mechanical component, electromagnetic noise in the leads connecting the strain gauges to the instrumentation mounted in the calibration rig, particular mechanical problems. Nevertheless it is shown that for this implant there was a good correlation between
the two sets of data

The improvement made by the new formula is visible, in this load case, at the strain gauge number two where there is a better approximation between the two values and where the old formula goes much below. Previously, with the old formula, the difference between these two results was actually explained by noise in the measurement by this strain gauge.

The new formula gives a slightly better correlation with what happens in reality making possible a better comparison between the data and, probably, a smoother migration from the algorithms developed with FEA data to algorithms using the experimental data from the calibration rig.

8.2 Matrix Method

These strains give the supposed result of the real strain gauges if the implant was built without imperfections and if there were no other mechanical or/and electromagnetic errors influencing the results.

Once that is a pure set of data not liable to the referred errors, it stands as the ideal for conceptualizing, test ideas and get a first prediction of the behavior of the strains gauges.

It is intended that with this data it would be possible to develop some algorithms to then be tested and adapted to the real implant.

During the research project development, an idea had emerged that was based in data of previous designs of the implant. The analysis was done using Microsoft Excel that provided statistical, algebraic and graphical tools and also support for creating applications based in the spreadsheets and Functions or/ and Macros to create more complex algorithms. Some of that work could be reused to point to new data for this new design.

A new spreadsheet was created and matrix inserted that compiled information according to the formulas:

\[ \text{Axial Force} = \text{Sum}(\text{strain gauge sensitivities}) \]

\[ \text{Bending Moment over Medio Lateral Axis} = \text{Sum} \ (\text{strain gauge sensitivity} \times \text{distance from strain gauge to origin over Antero-Posterior axis}) \]

\[ \text{Bending Moment over Antero-Posterior Axis} = \text{Sum} \ (\text{strain gauge sensitivity} \times \text{distance from strain gauge to origin over Medio Lateral axis}) \]

\[ \text{Shear Force over Antero-Posterior Axis} = \text{Sum} \ (\text{strain gauge sensitivity} \times \sin(\text{angle of strain gauge})) \]
Shear Force over Medio Lateral Axis = Sum (strain gauge sensitivity * cos(angle of strain gauge))

These formulas should define with some accuracy the forces applied in implant considering the strain gauge sensitivities.

8.3 Conclusion

Although there is no great improvement on the consonance of the new data with the new formula for strain seems to result in a better approximation to the practical data from the calibration rig.

Nevertheless, there is not enough data from the FEA to evaluate if the procedure may in fact, be used. As with some guess points it has been working, some data from the extremes gives incomprehensible results. It was decided to request a new set of data with this method to continue the work.

This work required a more deep study in mechanical principles also showing that this traineeship would not be possible without the help of other engineers versed in other subjects working in this project.
9 Conclusions and Future Work

This traineeship pursued mainly three goals, development of a front-end application, modification of already existing software and study of Finite Element Analisys data for algorithm development.

The main application finds itself in a last phase of tests. Some lack of performance of the LabVIEW platform should be compensated by more powerful computer hardware.

The application not only fulfilled the requirements as also added interesting extra functionalities for the user. The components from LabVIEW platform were in this case fundamental. Developing such components in conventional textual language would be cumbersome. For the success of this application, as having the referred characteristics, it was worthwhile the technological analyses. It was that discussion that finally identified the advantages that one could take from the used platform. Nevertheless, there where disadvantages and risks identified, a development plan was designed to diminish the magnitude of the most critical dangers.

Some of the dangers were inherit from the visual programming paradigm, like the known problem of lack of extensibility in these systems. Others were directly from LabVIEW, like the know problem of lack of flexibility in creating a responsive interface. These dangers were later shown in practice where better specification and structural organization were fundamental.

Visual programming is not, in fact new to the computer science world, although a discussion still continues if it actually brings better results to software development. In this case it is not possible to say that the application owes its success to the programming paradigm. It was specifically LabVIEW that provided useful tools to develop the application.

Nevertheless, the G language of LabVIEW is far from being so powerful as other textual languages.

In a first instance there is the lack of control over some of the software components as it was the case of the tab control that does not allow to, programmatically, change it to display a new page.

Also the manipulation of the data over the application is not so easy as data flows to be operated. Per example it has the equivalent to a “For” cycle but not the equivalent to the “continue” statement to jump the rest of the execution of one cycle. This is identified as lack of support for “ill-structured” diagrams [MEN98]. Ill-structured languages, like C, allow flexibility in programming. A well-structured language, like G, does not allow a new flow of data in a middle of a cycle.

The strength of LabVIEW seems to be more on its development environment, providing the software components, but also providing the development of straightforward diagrams easy to understand. It allows the quick development of simple applications include only some problems when the code gets larger if there is not a good structure for the application.

In the end, it still seems, that there is more to see from this language. The GOOP toolkit was not used but would allow to include Object Orientated Programming ideas in LabVIEW applications. This can be interesting to improve the structure of the application.
The version 7.0 of LabVIEW also may solve some of the flexibility problems with the inclusion of user events recognition.

The second goal of the traineeship was also based in LabVIEW. Here also combining technology with visual C++.

In this case, the previous software was studied until the needed changes became identified. Once this achieved modifications were made and the exact source of a problem became clear with these modifications. The situation was passed to the responsible engineer and response should come shortly. The previous knowledge of LabVIEW gained with the development of the front-end application was of clear importance.

The third goal was more theoretic than the previous goals. The development of the algorithm was not concluded but some improvements to the idea were made. Nevertheless the matrix method seems promising and more Finite Element Analyses data was requested to continue its exploration.

It should be referenced that this work uses electrotechnical and mechanical principles and therefore the contribution, with several discussions, from the rest of the research team was fundamental.

After the integration of the algorithm that determines the forces from the strain gauges sensitivities in the main application it will be ready for use.

The main application developed will help to determine experimentally, for the first time, the forces across the human knee joint.

In general the traineeship had a good impact on the trainee caused by the participation in a leading research project and a supporting department that helped to solve problems way beyond the professional ones.

The traineeship had also a good component of study of new technologies and their integration. Although it is quite easy to create diagrams with LabVIEW its full potentiality is yet to be discovered.

It was also a very positive experience to go through several engineering subjects more distant from the informatics engineering and be part of an engineering team contributing to evolution in orthopedics.
References and Bibliography


www.ni.com - National Instruments Website
APPENDIX A: Original List of Requirements

Project description (Project K-C2)  April 2003

‘Development of a user interface for capturing data on knee forces in real time.’

The main tasks are as follows:

1. To integrate the data interpretation algorithms (project K-C1) into a host environment for real-time data acquisition and interpretation. It should be possible to select calibration data for any one of several implants from a text file, for data logging.

   Possible host environments are: LabView, Visual Basic, MATLAB. The favourite is LabView since some of the code is already available in LabView, and the operation is visual and could be easily altered later if necessary. The data interpretation algorithms will be available from project K-C1 probably as C++ code. The implant calibration data files will be text files.

5. Write a user interface to log and plot the data with context-sensitive filenames in real time.

6. Graph plots will be needed to optionally view the results of one or more channels, in real time, and from saved data. Each acquisition will require a filename with fixed format having the patient’s number, the activity code letter, and the session number.

7. The Mk4.exe program for capturing data will need to be modified to decode certain other data.

8. Decode the timing of an external event to enable synchronisation with kinematic data from a motion analysis system.

The details of the work outlined above will be as follows:

1. Reading, selecting, and saving data

   Forces and positions from each knee compartment will be determined by a data interpretation algorithm (project K-C1). To provide interim data for the user interface, a file with representative force and position data from one compartment is available (sample knee data.xls). This has 1000 samples (taken at 10ms intervals) of the 6 channels to be displayed and logged for each compartment (6 from the left and 6 from the right knee compartment). Data will continuously become available at 100Hz.

   Data will be captured from any given patient (up to 8 patients) during sessions in a gait laboratory. Generally, data will be streamed to the PC for up to 30 seconds (1000-3000 data records); 24 channels of strain data at an update rate of 100 Hz (mode 0). Use of different modes will capture data from a smaller number of channels at faster rates. Gain selection will determine the sensitivity of the data to force: these two parameters are telemetered with the data and interpreted by Mk4.exe to decode the data accordingly.

   A menu system should enable the user to select or enter parameters during any given data logging session: patient number / name (linked to a LEFT or RIGHT knee which can only be
changed with a password), session number and date, activity name and letter code, patient mass (kg), patient self-assessment rating. The data should be selectively logged with context-sensitive filenames (of the form k1-02A for patient 1, session 2, activity code A). The user should be able to select either a pre-set acquisition time in seconds or to start and stop the acquisition on a keystroke.

The data for each compartment should be saved in the format of sample knee data.xls, where the channels are:

- **AXF**: axial force (+ve in compression)
- **APD**: antero-posterior distance from tibial tray baseline (anterior +ve)
- **MLD**: medio-lateral distance from the centreline (+ve away from centreline)
- **APS**: antero-posterior shear force (anterior +ve)
- **MLS**: medio-lateral shear force (+ve away from centreline)
- **AXT**: axial torque (clockwise +ve for LEFT compartment and –ve for RIGHT compartment)

LEFT and RIGHT compartments will map to MEDIAL or LATERAL respectively for a RIGHT knee; the opposite for a LEFT knee. The knee origin is the intersection between the centreline and the posterior edge of the tibial tray. The measurements are made with respect to a TIBIAL axis system.

### 2 Plotting data

The data should be selectively plotted in real time. The system should allow for different combinations of plots as follows:

1. Combined AXF, APS, MLS v. time (LEFT & RIGHT plots)
2. Resultant force and angle in either frontal or sagittal planes v. time (LEFT & RIGHT plots)
3. Combined MLD, APD v. time (LEFT & RIGHT plots)
4. Resultant force and angle in either frontal or sagittal planes v. time (LEFT & RIGHT plots)
5. MLD v. APD (surface plot)

### Resultant force plots

Since the force systems produced by each compartment are linked by the femoral component, we can combine the data from the 2 compartments to produce a 'resultant' force system acting across the knee, and these should be plotted as follows:

- Resultant AXF = AXF(L) + AXF(R)
- Resultant APS = APS(L) + APS(R)
- Resultant MLS = MLS(L) - MLS(R) for a LEFT knee, MLS(R) - MLS(L) for a RIGHT knee

### 3 Synchronizing the system with a motion analysis system
During patient trials, the patients will undertake activities in a gait laboratory, and kinematic data will be recorded by a motion analysis system in real time, on a separate computer. This data will be post-processed to determine certain features such as knee flexion angle, and this data will need to be incorporated into the telemetry database. To enable synchronisation between the 2 systems, an external event will be provided at the receiver (initiated either by the external system or manually by the operator on the receiver box), and encoded as for the patient box event (only one event is allowed; either at the patient box or at the receiver box). The revised Mk4.exe program will decode this event and its timing, and the user interface should provide a mark in the saved data and on saved plots.
APPENDIX B: Principal Stresses

In Diagram 1 we have shown a structure element with both normal axial stresses and shear stresses acting on the element. We remember at this point that for static equilibrium the shear stresses \( \tau_{xy} \) and \( \tau_{yx} \) must be equal in magnitude.

In Diagram 2 we have shown the structure element with a plane cut through it at angle \( \theta \). Acting on this plane will be both an axial stress \( \sigma_x \) and a shear stress \( \tau \), as shown in Diagram 2a. We would like to write relationships which allow us to calculate the value of these two stress for any arbitrary plane section.

In Diagram 2b, we have shown a triangular element with axial and shear stresses shown. If we multiply these stresses by the appropriate areas, we have the forces on each surface. We may then apply static equilibrium conditions and write the equilibrium equations. Before we do so, we need to establish a sign convention as follows:

1. Tensile Stress will be considered positive, and Compressive Stresses will be considered negative.

2. The Shear Stress will be considered positive when a pair of shear stress acting on opposite sides of the element produce a counterclockwise torque (couple). (Some text use the opposite direction for the positive shear stress. This changes a sign in several equations, so we must be somewhat careful of signs when working problems and examples.)

3. The incline plane angle will be measure from the vertical, counterclockwise to the plane. This will be the positive direction for the angle.

* Kindly from University of Wisconsin-Stout
We now write the following equilibrium equations:

Sum of Forces

\[- \sigma_x (A \cos \theta) - \tau_{xy} (A \sin \theta) - \tau_\theta A (\cos \theta) + \sigma_\theta A (\sin \theta) = 0\]

Sum of Forces_y

\[- \sigma_y (A \sin \theta) + \tau_{xy} (A \cos \theta) - \tau_\theta A (\cos \theta) + \sigma_\theta A (\sin \theta) = 0\]

where we have used that face that magnitude of \(\tau_{xy}\) = magnitude of \(\tau_{yx}\).

The two equation above may be solved for two "unknowns". In this case we solve for \(\sigma_\theta\) and \(\tau_\theta\); the stresses acting on the incline plane shown in Diagram 2. The details of solving these two simultaneous equations involve a number of trigonometric identities and some extended algebraic manipulations, and will not be presented. The results of this process are as follows:

\[\sigma_\theta = (\sigma_x + \sigma_y)/2 + [(\sigma_x - \sigma_y)/2] (\cos 2\theta) + \tau_{xy} (\sin 2\theta)\]
\[\tau_\theta = -[\sigma_x - \sigma_y]/2 (\sin 2\theta) + \tau_{xy} (\cos 2\theta)\]

The equations may be referred to as transformation equations. In the above equations, it is clear that we will may have both maximum and minimum stress values. The maximum and minimum normal axial stresses are known as the Principal Stresses, and the planes at which they occur are known as the Principal Planes. At the principal planes, where the axial stress is a maximum or minimum, the shear stress will be zero. The value of the principal(maximum/minimum) stresses are given by:

\[\sigma_{\text{max}} = \frac{\sigma_x + \sigma_y}{2} + \sqrt{\left[\frac{(\sigma_x - \sigma_y)}{2}\right]^2 + \tau_{xy}^2}\]
\[\sigma_{\text{min}} = \frac{\sigma_x + \sigma_y}{2} - \sqrt{\left[\frac{(\sigma_x - \sigma_y)}{2}\right]^2 + \tau_{xy}^2}\]
\[\tau_{\text{max}} = \pm \sqrt{\left[\frac{(\sigma_x - \sigma_y)}{2}\right]^2 + \tau_{xy}^2}\]

and

\[\tan 2\theta = \frac{\tau_{xy}}{(\sigma_x - \sigma_y)/2}\]

for the principal plane. The planes for maximum shear stress vary by 45° from the principal planes.
The principal stresses may also be related as follows:

\[ \sigma_{\text{max}} = \frac{\sigma_x + \sigma_y}{2} + r_{\text{max}} \]
\[ \sigma_{\text{min}} = \frac{\sigma_x + \sigma_y}{2} - r_{\text{max}} \]
\[ r_{\text{max}} = \frac{\sigma_{\text{max}} - \sigma_{\text{min}}}{2} \]
APPENDIX C: Strain Gauges Basics

What Is Strain?

Strain is the amount of deformation of a body due to an applied force. More specifically, strain (\(e\)) is defined as the fractional change in length, as shown in Figure 1 below.

\[
\varepsilon = \frac{\Delta L}{L}
\]

**Figure 1. Definition of Strain**

Strain can be positive (tensile) or negative (compressive). Although dimensionless, strain is sometimes expressed in units such as in./in. or mm/mm. In practice, the magnitude of measured strain is very small. Therefore, strain is often expressed as microstrain (\(\mu e\)), which is \(e \times 10^{-6}\).

When a bar is strained with a uniaxial force, as in Figure 1, a phenomenon known as Poisson Strain causes the girth of the bar, D, to contract in the transverse, or perpendicular, direction. The magnitude of this transverse contraction is a material property indicated by its Poisson's Ratio. The Poisson's Ratio \(n\) of a material is defined as the negative ratio of the strain in the transverse direction (perpendicular to the force) to the strain in the axial direction (parallel to the force), or \(n = \varepsilon_T / \varepsilon\). Poisson's Ratio for steel, for example, ranges from 0.25 to 0.3.

The Strain Gauge

While there are several methods of measuring strain, the most common is with a strain gauge, a device whose electrical resistance varies in proportion to the amount of strain in the device. The most widely used gauge is the bonded metallic strain gauge.

The metallic strain gauge consists of a very fine wire or, more commonly, metallic foil arranged in a grid pattern. The grid pattern maximizes the amount of metallic wire or foil subject to strain in the parallel direction (Figure 2). The cross sectional area of the grid is minimized to reduce the effect of shear strain and Poisson Strain. The grid is bonded to a thin backing, called the carrier, which is attached directly to the test specimen. Therefore, the strain experienced by the test specimen is transferred directly to the strain gauge, which

* A tutorial of National Instruments
responds with a linear change in electrical resistance. Strain gauges are available commercially with nominal resistance values from 30 to 3000 Ω, with 120, 350, and 1000 Ω being the most common values.

![Figure 2. Bonded Metallic Strain Gauge](image)

It is very important that the strain gauge be properly mounted onto the test specimen so that the strain is accurately transferred from the test specimen, though the adhesive and strain gauge backing, to the foil itself.

A fundamental parameter of the strain gauge is its sensitivity to strain, expressed quantitatively as the gauge factor (GF). Gauge factor is defined as the ratio of fractional change in electrical resistance to the fractional change in length (strain):

$$GF = \frac{\Delta R / R}{\Delta L / L} = \frac{\Delta R / R}{\varepsilon}$$

The Gauge Factor for metallic strain gauges is typically around 2.

**Strain Gauge Measurement**

In practice, the strain measurements rarely involve quantities larger than a few millistrain($e \times 10^{-3}$). Therefore, to measure the strain requires accurate measurement of very small changes in resistance. For example, suppose a test specimen undergoes a strain of 500 mc. A strain gauge with a gauge factor of 2 will exhibit a change in electrical resistance of only 2 ($500 \times 10^{-6}$) = 0.1%. For a 120 W gauge, this is a change of only 0.12 W.

To measure such small changes in resistance, strain gauges are almost always used in a bridge configuration with a voltage excitation source. The general Wheatstone bridge, illustrated below, consists of four resistive arms with an excitation voltage, $V_{EX}$, that is applied across the bridge.
The output voltage of the bridge, $V_O$, will be equal to:

$$V_O = \left[ \frac{R_3}{R_3 + R_1} - \frac{R_2}{R_2 + R_3} \right] \cdot V_{EX}$$

From this equation, it is apparent that when $R_1/R_2 = R_4/R_3$, the voltage output $V_O$ will be zero. Under these conditions, the bridge is said to be balanced. Any change in resistance in any arm of the bridge will result in a nonzero output voltage.

Therefore, if we replace $R_4$ in Figure 3 with an active strain gauge, any changes in the strain gauge resistance will unbalance the bridge and produce a nonzero output voltage. If the nominal resistance of the strain gauge is designated as $R_G$, then the strain-induced change in resistance, $\Delta R$, can be expressed as $\Delta R = R_G \cdot GF \cdot \varepsilon$. Assuming that $R_1 = R_2$ and $R_3 = R_G$, the bridge equation above can be rewritten to express $V_O/V_{EX}$ as a function of strain (see Figure 4). Note the presence of the $1/(1+GF \cdot \varepsilon/2)$ term that indicates the nonlinearity of the quarter-bridge output with respect to strain.

$$\frac{V_O}{V_{EX}} = \frac{GF \cdot \varepsilon}{4} \left( \frac{1}{1 + GF \cdot \frac{\varepsilon}{2}} \right)$$

Figure 4. Quarter-Bridge Circuit

Ideally, we would like the resistance of the strain gauge to change only in response to applied strain. However, strain gauge material, as well as the specimen material to which the gauge is applied, will also respond to changes in temperature. Strain gauge manufacturers
attempt to minimize sensitivity to temperature by processing the gauge material to compensate for the thermal expansion of the specimen material for which the gauge is intended. While compensated gauges reduce the thermal sensitivity, they do not totally remove it.

By using two strain gauges in the bridge, the effect of temperature can be further minimized. For example, Figure 5 illustrates a strain gauge configuration where one gauge is active \((R_G + DR)\), and a second gauge is placed transverse to the applied strain. Therefore, the strain has little effect on the second gauge, called the dummy gauge. However, any changes in temperature will affect both gauges in the same way. Because the temperature changes are identical in the two gauges, the ratio of their resistance does not change, the voltage \(V_O\) does not change, and the effects of the temperature change are minimized.

![Diagram](image)

**Figure 5. Use of Dummy Gauge to Eliminate Temperature Effects**

The sensitivity of the bridge to strain can be doubled by making both gauges active in a half-bridge configuration. For example, Figure 6 illustrates a bending beam application with one bridge mounted in tension \((R_G + DR)\) and the other mounted in compression \((R_G - DR)\). This half-bridge configuration, whose circuit diagram is also illustrated in Figure 6, yields an output voltage that is linear and approximately doubles the output of the quarter-bridge circuit.

![Diagram](image)

**Figure 6. Half-Bridge Circuit**

Finally, you can further increase the sensitivity of the circuit by making all four of the arms of the bridge active strain gauges in a full-bridge configuration. The full-bridge circuit is shown in Figure 7.

65
The equations given here for the Wheatstone bridge circuits assume an initially balanced bridge that generates zero output when no strain is applied. In practice however, resistance tolerances and strain induced by gauge application will generate some initial offset voltage. This initial offset voltage is typically handled in two ways. First, you can use a special offset-nulling, or balancing, circuit to adjust the resistance in the bridge to rebalance the bridge to zero output. Alternatively, you can measure the initial unstrained output of the circuit and compensate in software.

The equations given above for quarter, half, and full-bridge strain gauge configurations assume that the lead wire resistance is negligible. While ignoring the lead resistances may be beneficial to understanding the basics of strain gauge measurements, doing so in practice can be a major source of error. For example, consider the 2-wire connection of a strain gauge shown in Figure 8a. Suppose each lead wire connected to the strain gauge is 15 m long with lead resistance \( R_L \) equal to 1 W. Therefore, the lead resistance adds 2 W of resistance to that arm of the bridge. Besides adding an offset error, the lead resistance also desensitizes the output of the bridge.

You can compensate for this error by measuring the lead resistance \( R_L \) and accounting for it in the strain calculations. However, a more difficult problem arises from changes in the lead resistance due to temperature fluctuations. Given typical temperature coefficients for copper wire, a slight change in temperature can generate a measurement error of several me.

Using a 3-wire connection can eliminate the effects of variable lead wire resistance because the lead resistances affect adjacent legs of the bridge. As seen in Figure 8b, changes in lead wire resistance, \( R_2 \), do not change the ratio of the bridge legs \( R_3 \) and \( R_G \). Therefore, any changes in resistance due to temperature cancel each other.
Figure 8. 2-Wire and 3-Wire Connections of Quarter-Bridge Circuit