Study of behaviour of Biomechanical System in indented Articular Cartilage on a cellular level

Ramezanian Kalahroudi, M.

MPhil Report

Title
Study of behaviour of Biomechanical System in indented Articular Cartilage on a cellular level

Submitted by

Mohammad Ramezanian Kalahroudi

Director of Studies
Dr Philip Trwoga

Second Supervisors
Dr Ian Locke, Dr Andrew Afoke

Electronic and Computer Science Research Group

September 2018

Contents

1 Introduction

Abstract

The study of biomechanical systems is of great interest to researches due to diverse applications in the medical sector. This study focuses on design and implementation of a mechanical device, a novel dual axis construct simulator (DACS) for in vitro studies on immortalised chondrocytes. DACS will help with experimental measurements of mechanical properties of Articular Cartilage (AC) on a cellular level and the relationship among cellular, pericellular and extracellular deformation in AC. Details provided in this research mainly focuses on software and hardware development processes and challenges involved. There will a brief introduction about biomechanical behaviour of the cartilage and DACS impact on future studies. This will be followed by description of LCMPilot, an experimental software and challenges involved to develop and control DACS in an automated setup.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my Supervisor Dr. Philip Trwoga for the continuous support of my MPhil study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

My sincere thanks also go to Dr. Ian Lock and Dr. Andrew Afoke, who provided me an opportunity to join their team, and who gave access to the laboratory and research facilities. Without their precious support, it would not be possible to conduct this research.

Author's Declaration

I declare that all the material contained in this thesis is my own work.

1.1.1 Problem State

Osteoarthritis is clinically associated with the functional breakdown of AC. This commonly shared with other form of arthritis. The synovial joint is a mechanical system; during its operation AC is exposed to a wide range of motion under load. AC undergoes cyclic load and the mechanical forces can change its structure and composition; this results in change of biomechanical behaviour of the cartilage. AC unique structure help to reduce peak forces during normal joint motion. Studies show that the mechanical forces are essential for maintenance of the articular cartilage, however excessive mechanical forces can cause degeneration of the tissue (F. Ghadially et al, 1983, L. Eichelberger et al, 1952, J. P. Paul et al, 1976, C. G. Armstrong et al, 1979). It is essential to identify a regime optimised to enhance and maintain the AC structure.

Articular cartilage is comprised of the only cell called chondrocytes and extracellular matrix which consists of water, collagen fibres and proteoglycans. The chondrocytes synthesize the components of the cartilage and maintain its biomechanical properties. Collagen fibres are responsible for supporting tensile stress, they form a fibre network that help to restrain the swelling pressure caused by proteoglycans, the negatively charged macromolecules in the cartilage. This is how the cartilage is provided with its resilience and loading capacity. Before the use of DACS the study of the structure of articular cartilage was not possible when mechanical forces were being applied to the tissue. DACS allows the study of the structure of AC in its natural hydrated state.

1.1.2 Research Objectives

One of the objectives of this study is to understand the behaviour of tissues under cyclic loading and shearing stress and its effect on AC development, destruction and repair. We aim to utilise an established type I collagen gel culture system to conduct our studies on C-20/A4 chondrocytes.

In particular the aims of this project are: i. Development of a novel dual axis construct simulator (DACS) for in vitro studies on immortalised chondrocytes C-20/A4 and to verify and extend previous observations reported in literature, ii. Demonstrate the potential effect of long-term dynamic mechanical simulation on chondrocytes extracellular matrix synthesis and viability, iii. Development of proof-of-principle experiments in vitro, validating novel approach to closely mimic in vivo conditions, iiii. Delivery of novel approach in finding the required loading regime to improve AC repair process.

2 Literature Review

2.1 Articular Cartilage

Several reviews have been published on the general mechanical properties and design features of articular cartilage (F. Guilak, 1997, Kääb, 1998, W. Herzog, 1998). Findings confirm the impact of repetitive loading on metabolic activities in AC (URBAN, 1994). AC is an avascular (lack of blood vessels) and aneural (no nervous) tissue type found in the synovial joints such as knee. Since AC is avascular it has limited ability to regenerate and to repair itself into fully functional tissue. Therefore, mechanical stimulation of AC is essential for the maintenance of this tissue.

This specialised connective tissue acts as a junction of two bones in human knee, the Femoral Condyle (FC) and Tibial Plateau (TP) (Rami and Simo, 2011, Wong and Sah, 2010). This type of bone articulation permits free motion in diarthrodial joints (Archer, 1994). During joint articulation cartilage is subjected to loading, shear, sliding and hydrostatic pressure (Hall, 1991). These are mechanical factors that affect cartilage metabolism (Nguyen et al., 2010). As highlighted above, this has motivated the design of several systems to closely simulate the mechanical loading that match the physiological loading conditions in knee joint.

However, these experiments produced contradictory results. For example, Di Federico et al. in a study of chondrocytes seeded in agarose gel, reports that during various experimental conditions (up to 72h loading), the chondrocyte viability was maintained above 90%. Following this result the study then suggests that the culture conditions and the mechanical loading regimes have no impact on cell viability or mechanical integrity of construct. In this study culture medium had to be changed manually (Di Federico et al., 2014).

These studies also often capture certain aspect of a knee motion in their experiments. For example Frank et al., in development of a dual axis simulation, only isolates the effect of dynamic or static shear in 24h experiments with statically compressed control. Although the explants in this experiment was maintained in culture media, little is disclosed as how this was maintained and whether cell viability was assessed. Although this study was only conducted on identifying the shear modulus of load bearing explants, the findings did demonstrate increase of dynamic stiffness and synthesis of proteoglycans (Frank et al., 2000).

Hence development of a novel simulator capable of applying both axial and shear loading is paramount to the success of the active knee joint simulation in this study.

AC also acts as dissipater and distributor of contact stresses during joint loading (Rami and Simo, 2011). It is compromised of the highly specialised and fibroblast-like chondrocytes and associated Extracellular Matrix (ECM). ECM components are produced and maintained by the chondrocytes and include water, type II collagen, and proteoglycans, with other non-collagenous proteins and glycoproteins present in lesser amount (Buckwalter, 1998, Buckwalter JA, 1988). The importance of collagen fibres for the mechanical function and integrity of cartilage has been demonstrated experimentally (R.A. Bank, 2000, Maroudas, 1976, J. Mizrahi, 1986, N. Verzijl, 2002).

2.1.1 Cell Signalling

Chondrocytes are responsible for maintenance of ECM through a complex interplay of anabolic and catabolic stimuli. Their functionality is conducted by a complex signalling network. In osteoarthritis, a disruption in the anabolic and catabolic processes cause degeneration of the ECM and gradually followed by destruction of the joint (Guilak, F., et al., 2004). Modelling the signalling mechanisms is a complicated task for academic and industry researchers as they attempt to either block pro-inflammatory pathways that leads to cartilage degeneration or stimulate pro-growth pathways to compensate for the loss of ECM structure.

In the past, the modelling of signalling in chondrocytes was conducted in a traditional approach by modelling the effect of few well known stimuli on proteins, cell's phenotype and tissue development. In literature, there are references to the effect of stimuli such as IL la/b, Interleukin 1 alpha/beta, TNFa-Tumor Necrosis Factor alpha and Transforming Growth Factor alpha (TGFa) on proteins like Nuclear Factor kappa beta (NFkB) or ikb (R. Visse, et al., 2003, Goldring et al., 2007, Palmer et al., 2009, DeLise et al., 2000, Hartmann et al., 2000, Goldring et al., 2006, Roman-Blas, et al., 2006, Liacini et al., 2002). However deeper understanding of cell's mechanical structure is required to identify the therapeutic interventions.

Simple investigation of each biomechanical cascade independently although could be helpful it is not enough (Kitano, 2002, Boccaletti et al., 2006, Aggarwal et al., 2003). Ioannis et al., in a study on chondrocytes isolated from a single donor aimed to construct a chondrocytes specific signalling network based on proteomic data and existing knowledge of protein connectivity. The sample was placed in a 96 well plate and stimulated with single treatments of 78 stimuli.

17 key phosphoprotein signals were measured using xMAP technology. The study proposed a predictive model of chondrocytes signalling network. The findings revealed discrepancies with generic nature of canonical pathways in previous studies (Ioannis et al., 2011). This experiment is one of the early attempts made to construct an integrative model of chondrocytes signalling mechanism.

Although the results show that chondrocytes do not respond to all stimuli, however 18 cytokins are identified as having a clear effect on some of the 17 signals. This provides better insight into functional mechanism of chondrocytes and ECM structure integrity and how DACS could utilise the findings and evaluate data in future studies.

2.1.2 Articular Cartilage Morphology and Cartilage Matrix

There are three types of cartilage, hyaline, elastic and fibrocartilage. This categorisation is based on morphologic criteria, collagen (Types I and II) and elastic content. The most common type is hyaline, found as supportive tissue in the nose, ears or knee. As AC, Hyaline Cartilage (HG) covers the articular surfaces of bones in synovial joints and provides frictionless surface for articulation (Sophia Fox, 2009). AC contains a gelatinous ground substance called chondroitin sulfate (CS). Embedded within CS are collagen and elastic protein fibres. Together these components form the AC matrix that is flexible and also resistant to compression forces (Naumann A, 2002, Watanabe, 2015).

The integrity of AC depends on the functioning and mechanical simulation of chondrocytes. This is the cell that synthesise extracellular matrix and maintain tissue health. However mechanical loads do not only effect the cartilage matrix, but also have impact on cartilage cells known as Chondrocytes (CH) (Clements et al., 2001, Wu et al., 1999, J.Z. Wu, 2000, Jones et al., 1999). Helminen et al., has reviewed decades of research evidences and confirms that CH also responds to mechanical forces and is capable of cartilage remodelling (Helminen, 1987). Therefore mechanical environment of chondrocytes are considered to be important factor for joint health.
Chondrocyte deformation in response to mechanical loading is an important regulator of metabolic activity (Han et al., 2007). Chondrocytes are the only cells in cartilage. They produce and maintain the cartilage matrix.

2.1.3 Cartilage biphasic composition

This study will utilise the use of dynamic loading with indentor for in vitro experiments. The combination of mechanical and chemical in vitro conditions will have significant impact as it could lead to greater understanding of cartilage functional tissue engineering. Majority of researchers confirm that AC has a biphasic composition and that it can be described in two phases (Park et al., 2004). Fluid phase that consists of interstitial water and mobile ions and constitutes nearly 85% of AC total weight. This is an important determination of mechanical properties of AC and also contributes to the time dependent properties of the tissue (Wu et al., 1999, J.Z. Wu, 2000).

The second phase is referred to as the solid phase or solid matrix of AC and consists of collagen fibrils (60-80% of dry weight) and proteoglycans (20-40% of solid weight) as mentioned above (Michael D. Buschmann, 1995, Rami and Simo, 2011, Seifzadeh et al., 2012, Park et al., 2003). Collagen fibrils are responsible for cartilage tensile and dynamic comprehensive stiffness (Guilak and Mow, 2000, Korhonen and Herzog, 2008). Where proteoglycans are responsible for equilibrium properties during compression (Rami K. Korhonen, 2003, Halonen et al., 2013).
This complex behaviour introduces advance problems in finite element simulation. To simplify the problem researches work under various assumptions to idealise their models.

In a literature review Freutel et al., indicates that simple models (ie: for swelling) have been developed that consider the cartilage to be homogeneous (Freutel et al., 2014). Park et al., also suggests that AC is homogeneous through depth (Park et al., 2004). In a study by Mow et al., the same assumption was used to model ECM. However it was also highlighted that modifications are required to use the same technique to incorporate other physiological characteristics such as inhomogeneity of the cell (Guilak and Mow, 2000).

2.1.4 Cartilage structure and chondrocyte distribution

There is also evidence about the structure of AC and that it can be divide into four zones based on the arrangement of collagen fibril network (Rami and Simo, 2011, Halonen et al., 2013, Bi, 2006). The zones are known as Superficial, Middle, Deep and Calcified. This suggests that the structure of the collagen of AC is capable of exhibiting a zone-specific deformation that is dependent on the magnitude and type of load (Kääb, 1998, Korhonen and Herzog, 2008). In an experiment Clements et al. reveals that the first sign of load-induced CH death was greatest in superficial zone and then extended beyond the loaded area to other zones (Clements et al., 2001). As a result of this depth dependency, variations of biomechanical parameters has been observed in many experimental studies (R.M. Schinagl, 1997). In a similar study, Herzog reported that depletion of cell volume was first observed in superficial zone after increase of fibril stiffness (Korhonen and Herzog, 2008).
Other studies also shown similar variations of young modulus (measurement of stiffness in elastic material), aggregate modulus (measurement of material stiffness at equilibrium), Poisson's ratio (change of shape) and permeability with depth (Chen, 2001a, Chen, 2001b, Jurvelin, 1997, Treppo, 2000). However, only few studies are related to human AC (J.S. Jurvelin, 2003, Pfeiler et al., 2008, Shirazi and Shirazi-Adl, 2009, Sun et al., 2011) and to the best of our knowledge none of the studies measured all the above parameters for the same samples under both axial and shearing stress.

2.2 Mechanobiology and Physical Activity

2.2.1 Dynamic Compressive loading and cell viability

Now, the magnitude of load and frequency or duration of a loading pattern and its effect on the chondrocyte is what this study is designed to systematically determine. Several studies are conducted on the effect of cyclic load on cartilage explants sourced from human or animal materials such as bovine. These studies try to demonstrate whether and to what extent the frequency and magnitude of cyclic loading modulates the biosynthesis of cartilage and to evaluate chondrocytes viability under different loading patterns. A decline in chondrocyte numbers is associated with osteoarthritic cartilage in the literature. The decrease in cell viability is caused by mechanical load. This unique physiological changes to the cartilage matrix has been reported to be similar to what was observed in early stages of osteoarthritis (Chen et al., 2001, Lucchinetti et al., 2002, Sauerland et al., 2003).

Lucchinetti et al. experiment on mature cartilage explants confirms that one of the mechanisms for the initiation of cartilage degeneration is mechanically induced cell death. This observation shown the matrix damage appears to be higher in articular cartilage surface and within the superficial tangential zone (STZ). In this experiment the AC was subjected up to 72 hours repetitive loading and frequency regimes (Lucchinetti et al., 2002). Finding suggests that the cell death did not appear to be caused by apoptosis (programmed cell death or cell suicide).

This very phenomenon (apoptosis) is still in question (Aigner T, 2001, Blanco, 1998). In a study of canine explant under cyclic load, Chen et al., reported that necrosis occurrence was first observed and then followed by apoptosis (Chen et al., 2001). There is also evidence in the literature that apoptosis can be triggered by the products of dead or injured cells such as nitric oxide (Prince, 2015, Amin, 1998). Although this was not fully confirmed in Lucchinetti et al. experiment, potential causes for cell death was suggested to be: i. excessive mechanical loading, ii. Damage to the articular surface caused by the rough surface of the porous platen, iii. Zone dependency of chondrocytes (chondrocytes in STZ are more susceptible to injury).

Same results are echoed in an experiment conducted by Sauerlan et al. in a study of mature bovine AC explants that was exposed to different loading patterns. In this experiment the AC was subjected to frequent loading for up to 6 days with load frequency of 5, 10 or 20s followed by unloading period lasting 10, 100 or 1000s. The findings confirm the importance of the frequency of cyclic loading as a mechanical factor to control metabolic activities in chondrocytes. This experiment also highlights the impact of mechanical damage to generate an in vitro model of degenerative OA like cartilage (Sauerland et al., 2003).

Further research in this matter also revealed same observation in human body. The number of chondrocytes decreases in degenerative disease of articular cartilage known as Osteoarthritis (OA) most commonly seen in elderly people (Amin, 1998). Similar study reveales that OA cartilage had a higher proportion of apoptotic chondrocytes than did normal tissue (51% versus 11%; $P < 0.01$). This study further shown that chondrocytes in OA cartilage demonstrated morphological changes similar to of apoptosis and suggested that this mechanism of cell death plays an important role in new treatment strategies (Blanco, 1998).

Both clinical and experimental researches for potential Osteoarthritis (OA) therapy highlight the same possibility (G.E. Nugent-Derfus, 2006) to prevent AC destruction as observed in OA or improve AC repair process (Song J, 2014).

2.2.2 Mechanical testing geometries

To test mechanical properties of articular cartilage under load we can utilise three different measurement configurations: Confined, un-confined compression (R.M. Schinagl, 1997, L.P. Li, 1999, Korhonen et al., 2002, Park et al., 2003, Park et al., 2004, C.P. Neu, 2005) and indentation (Zhang et al., 1997, Korhonen et al., 2002, Seifzadeh et al., 2012). In an unconfined compression, a tissue sample is compressed between two smooth metallic plates to a predefined stress or strain. In this geometry the interstitial fluid flow out of the tissue only in lateral direction. In a confined compression a tissue is placed in a sealed chamber and compressed with a porous filter. In this geometry, the interstitial fluid flow is observed to be axially and through tissue surface into the filter.

As mentioned earlier, this study will use dynamic loading with indentor, similar setup to indentation geometry to measure AC response to compression. Indentation is the only compressive geometry that can be used in vivo or outside laboratory testing. In this setup fluid flow is possible in both axial and lateral directions.

In a normal walking regime, AC and meniscus will experience external compressive forces. Each knee joint contains an inner and outer meniscus (medial and lateral meniscus). These are C-shaped rubber like pads and act as load bearing component in knee joint (V.C. Mow, 2005). The meniscal cartilage sit on top of the TP and are in addition to the thin layer of cartilage (also known as articular cartilage) that covers the top of TP.

During knee movement FC articulation against TP also causes friction, a phenomenon known as shearing. The FC and TP cartilage articulation facilitates the biomechanical movement of joint (Buckley et al., 2010, Wong et al., 2010, Wong and Sah, 2010). This is in addition to the perpendicular or axial force applied during knee active motion. Shearing provides promising insight into biomechanical responses of AC to distinct loading regime and articulation (Nguyen et al., 2010).

Although the shearing properties of AC was observed in different studies as early as 1993 (W. Zhu, 1993), to this date studies are conducted under assumption that the loading platens are frictionless in unconfined and confined compression. This assumption is reasonable due to low friction coefficient of AC under dynamic load (Park et al., 2004).

However, other studies suggest that friction coefficient can be significantly affected by platen roughness ($p < 0.001$) (Nguyen et al., 2010). Nguyen et al., in a microscopic assessment of cartilage shear suggests that roughened counter-surface could modulate interaction with cartilage surface and provide a means to control cartilage shear and sliding. This study examined the isolated effect of shear on cartilage and chondrocytes metabolism by use of 8 roughened custom counter surface platens (Polished, Mildly rough and Rough). A smooth counter surface (Polished) was recommended to be suitable to mimic articulation tests with normal cartilage and rougher versions (Mildly rough and rough) could be used to mechanically induce high stiffness in cartilage, typical of what was observed during articulation of degenerative cartilage.

The degree of how this articulation effects the AC metabolism is discussed in a study by Wong et al.. The findings indicates that TP cartilage is generally thicker than FC and deforms and strains more axially and in shear than the FC cartilage. This is due to the difference in mechanical stiffness; FC stiffness is 1.5 to 2 times higher than TP cartilage (Wong and Sah, 2010). This provides better understanding of functional AC tissue and its behaviour under physiological load.

Direct mechanical forces have been applied to cartilage explants either statically (Kong, 2013) or dynamically (Di Federico et al., 2014, Yusoff et al., 2011, Frank et al., 2000, Moo et al., 2014) to mechanically damage cartilage with variable frequency in loading regimes. Many of these loading regimes were chosen to simulate more closely the conditions cartilage might experience during normal or pathological joint function. Findings suggest that frequency and magnitude of loads have important effect on metabolic activity of cell. The most common mode of loading in human lower limb joints is cyclic loading and followed by shearing. These joints are subjected to four million load cycles per year on average (Seedhom and Wallbridge, 1985). During knee active motion, AC experiences periods of recovery or relaxation (complete unloading) between each loading cycle and shearing.

This research aims to simulate the active motion of knee joints for further analysis of signals generated by mechanical stress. The developed system is suitable for a novel approach to a systematic investigation of the response of chondrocytes cell line to a complex physiological deformation profile. In situ conditions, these mechanical signals are then converted into biochemical events via intracellular mechanisms. Hence such simulations may also provide insight into the repair and destruction of cartilage (Blanco, 1998, Deschner J, 2003, G.E. Nugent-Derfus, 2006). A custom built apparatus was developed to simulate the compressive and shear forces and loading regimes present in the knee joint.

A previously developed type I collagen gel will be used as the basic construct populated with the phenotypically stable chondrocyte cell line C-20/A4 (A. Petsa, 2004). The gel is highly hydrated and may not reflect the true conditions in vivo, however the use of in vitro systems such as immortalised cell lines over primary alternative offers important advantages over animal studies. In vitro, models allow the study of cellular response to loading more directly. This cell line is able to generate type II collagen and proteoglycans, the predominant ECM components.

The use of the culture system will also enable us to study the functional aspects of these cells with regard to their ability to modify the ECM. Gels are subjected to cyclic loading under variable regimes. The load cycles are controlled via a multi-threaded software and can be defined within sets of multiple sessions with resting and sleep periods for media flow. The use of in vitro collagen matrices to mimic an in vivo cellular environment is a quite popular approach and is broadening our understanding of cellular processes and cell - ECM interactions (O'Connor et al., 2001).

In a compression setup the equilibrium response of AC can also be measured. This can be significantly different from other measurement configurations. Korhenen et al. in a study in comparison of equilibrium response of AC revealed high variance in young's modulus of the in situ indentation test than those obtained from the in vitro unconfined and confined tests. The finding suggests that this discrepancy may depend on indentor size in use (Korhonen et al., 2002). This is later echoed by Lock et al. in a study, where the effect of indenter shape on creep curve test of a collagen gel was monitored. The results from this study indicated that the shape of indenter plays an important role in producing rapid displacement. This is an important finding in development of a simulator as a mechanism to control water content of the gel matrix during cyclic load. It could also have significance in vivo since the fluid movement within AC matrix is important for lubrication and nutrition (Locke et al., 2010).

The findings related to the shape of indentor and its counter surface roughness is reflected in the design of DACS where the DACS's indentor and its roughness can be adjusted to verify and extend previous findings in the literature.

The development of this simulator provides not only a novel system for in vitro studies on molecular regulations of chondrocytes but also a significant testing ground for development of targeted pharmacologic means and bio-mechanical assessments to regulate matrix production and to potentially regenerate cartilage. This will be a hybrid approach to combine series of important experiment configurations such as magnitude of load, frequency of load and recovery periods between each cycle.

This research is based on preliminary work that was undertaken in the study of collagen gel cultures conducted by Petsa et al. at University of Westminster. Under this study a set of type I collagen gels were investigated for prolonged culture times under static compression and gel stiffness was measured. The test gel was seeded with C-20/A4 chondrocytes cells (A. Petsa, 2004). In a 16 days period the resistance to compression in test gel seeded with chondrocytes increased over time from day 8 onwards. The load application had contributed to the synthetic activity of the chondrocytes and caused the production of modified extracellular matrix containing type II collagen, elastin and low levels of proteoglycans following as little as 1 day in culture. The findings also indicate that although necrotic cell death was low during the entire period however the levels of apoptosis were increased. This maybe a result of culture conditions, such as reduced oxygen or nutrition level (Wernike et al., 2008). There is also increased evidence that chondrocytes apoptosis have effective role in cartilage development, aging (Horton Jr et al., 1998) and diseases (Horton Jr et al., 1998). This experiment further demonstrated that chondrocytes in an in-vitro culture are capable of the remodelling and development of collagen gel matrix into a material similar to that of the in vivo environment of human articular chondrocytes.

3. Materials & Software Methodology

This section is dedicated to discussion about functional and non-functional requirements of developing DACS and challenges involved in implementing the features in final design. This section is broken to four subsections to list functional and non-functional requirements of developing DACS and LCMPilot, followed by discussion about use cases that define the interactions between LCMPilot and DACS. Finally, this chapter finishes with discussion about apparatus development.

The main challenge was to develop a multi-threaded application that can run set of coordinated tasks automatically.

3.1 Non-Functional Requirements

This section will provide detailed information about list of hardware and tools that were used to develop DACS.

R1 **Media Pump**
R1.1 Two micro range of peristaltic pumps model 100.005.012.030/4 with 4 rollers. This motor is 12v DC powered. This will pump the media fluid into and out of petri dish.
R1.2 Two tubes to isolate the liquid feed.
R1.3 Two 5V Single Pole DIL Reed Relays.

R2 **Materials for Testing**
R2.1 Wet sponge
R2.2 Agarose Gel

R3 **DACS (a custom designed apparatus)**
R3.1 Two hybrid 5v stepping motors
R3.2 PCI230 multi-functional analogue and digital input and output amplicon board.
R3.3 A strain gauge load cell LQB 630, thin film load cell with load range up to 50g.
R3.4 A Signal Amplification Mantacourt SGA/D signal conditioner with gains up to 30.30mV to amplify the strain gauge output.
R3.5 A switch button for emergency process halt.
R3.6 Petri dish with 5ml working volume to hold test samples
R3.7 Indentor in different size and shape.
R3.8 A plane ended indentor with possibility to mount indentors of different cross section.

R4 **LCMPilot (the controller software)**
R4.1 Visual Studio 2010
R4.1 C# programing language
R4.2 Mongo DB
R4.3 JavaScript
R4.4 Microsoft Excel

3.2 Functional Requirements

R5. The system should be able to register the PCI card for signalling the DACS components.
R6. The system should be able to control both vertical and horizontal motors.
R7. The system should be able to control the pump motor to pump the media feeds in and out of petri dish where the testing materials are placed, while DACS is in resting period.
R8. The system should provide settings feature to enable researchers to run automated cycles in one experiment.
R9. The system should be able to perform all above tasks in multiple coordinated cycles. This must be achieved by use of a technique called managed threading.
R10. The system should be able to store strain data in CSV format.
R11. The system should be able to send data to a remote server in JSON format.
R12. The system should provide an interactive console based application to configure and run an automated experiment.

3.3 Use cases

This section defines the interaction between actors and the system, actors in this case can be researchers or DACS.

The following tables represent the use case documentation of LCMPilot.

Use Case ID:	1
Use Case Name:	R5

Actor:	LCMPilot
Description:	Register the PCI card to signal and receive feedback from hardware components.
Preconditions:	Card is installed properly and the drivers are installed. We are using Amplicon C# API to register and communicate with the PCI card.
Post-conditions:	Once card registered, get the initial/current voltage values
Priority:	1
Frequency of Use:	When LCMPilot console application loads first.
Normal Course of Events:	1. Console app loads 2. Looks for connected PCI boards 3. Register the PCI card 4. Get current voltage from strain gauge 5. Asks for experiment configuration setup
Exceptions:	1. Software Loads, but cannot find the card. 2. Software Loads, but PCI board is busy or non-responsive.

Includes:	R6, R7
Assumptions:	The PCI card is functional and drivers are installed.
Notes and Issues:	This board have 52 ports and some are damaged and no longer work.

Use Case ID:	2
Use Case Name:	R6

Actor:	LCMPilot
Description:	Wind and wind back vertical and horizontal motors. These motors control the sheering and indentation features of DACS.
Preconditions:	Register PCI card, initiate the LCM threads, the threads are for controlling the vertical/horizontal motors and the media flow.
Post-conditions:	Do not stop the main thread (that's the application thread) Initiate the LCM motor tasks.
Priority:	2
Frequency of Use:	In each cycle
Normal Course of Events:	1. Move vertical motor 2. Move horizontal motor 3. Wind back the vertical motor 4. Wind back the horizontal motor 5. Store LCM strain data in an array
Exceptions:	PCI ports may dis-function, should change the port. System failure, the halt button will hard reset the system.
Includes:	
Assumptions:	
Notes and Issues:	This requires custom threading described in the following section.

Use Case ID:	1
Use Case Name:	R7

Actor:	LCMPilot
Description:	Run the media flow to and from the petri dish
Preconditions:	Session cycles are completed and system is in resting period.
Post-conditions:	Media flow completed and pump task is stopped before a new session cycle starts.
Priority:	2
Frequency of Use:	During each cycle
Normal Course of Events:	1. Initiate the Pump task 2. This task will run only if motors and session states are at rest.
Exceptions:	
Includes:	
Assumptions:	
Notes and Issues:	This requires custom threading technique described in the following section.

Use Case ID:	1
Use Case Name:	R8

Actor:	Researcher
Description:	Researchers can configure an experiment cycle that runs automatically for X number of times.
Preconditions:	PCI card is registered, and current voltage feedback from strain gauge is available. Internet connection is required.
Post-conditions:	Should save the strain gauge data in CSV format and send a JSON version to a remote server.
Priority:	3
Frequency of Use:	Once, when the application loads first time.
Normal Course of Events:	1. Ask for following information to setup a cyclic experiment: a. Number of sessions b. Load cycle in each session c. Load unit in grams d. Sheer depth e. Resting Period in hours f. Media cycle in a specified minutes interval g. Author name 2. Use author name to save experiment data in a remote server in json format.
Exceptions:	Remote server connection may timeout.
Includes:	
Assumptions:	
Notes and Issues:	In server connection timeout, a local copy of experiment data will be saved in CSV format.

Use Case ID:	1
Use Case Name:	R9

Actor:	LCMPilot
Description:	Each cycle will run the above tasks in multiple times.
Preconditions:	A custom threading technique is required to run the tasks in a cyclic order.
Post-conditions:	
Priority:	
Frequency of Use:	
Normal Course of Events:	Initiate managed multithreading to run pumping tasks and motion motors.
Exceptions:	
Includes:	
Assumptions:	
Notes and Issues:	Coordinating the work of multiple threads and handling threads that block.

Use Case ID:	1
Use Case Name:	R10

Actor:	LCMPilot
Description:	Save the LCM data in a CSV format.
Preconditions:	Trigger only when the session cycle is completed.
Post-conditions:	
Priority:	4
Frequency of Use:	Once, at the end of a session or experiment only.
Normal Course of Events:	1. Check for pumping tasks state, must be completed. 2. Check for the session state, must reach the end of cycles. 3. Save LCM data in CSV format locally.
Exceptions:	
Includes:	
Assumptions:	
Notes and Issues:	Permission error while saving the file. Always save on desktop as logged-in user have access to desktop.

Use Case ID:	1
Use Case Name:	R11

Actor:	LCMPilot
Description:	Send the LCM data to a remote server in json format
Preconditions:	Trigger only once the CSV file has been saved.
Post-conditions:	
Priority:	5
Frequency of Use:	Once, at the end of each session.
Normal Course of Events:	1. Convert the LCM data to json format 2. Send the json data to a remote server.
Exceptions:	
Includes:	
Assumptions:	
Notes and Issues:	Possible issues with internet connection, out of scope of LCMPilot to address an automated troubleshooting.

3.4 Apparatus and LCM Development

It is well stablished that continues active motion of a knee joint should accelerate the healing of AC (Salter, 1989, Helminen, 1987, G.E. Nugent-Derfus, 2006, Shirazi and Shirazi-Adl, 2009). With recent advances in science and evidences in literatures, it is well established that excessive mechanical stress is known to initiate cartilage destruction as seen in OA. On the other hand moderate compression and load regimes may increase number of chondrocytes. This is required for production and maintenance of ECM components (G.E. Nugent-Derfus, 2006).

To successfully simulate active motion of a knee joint, a custom built apparatus was developed (*Figure 1*). The preliminary design of this device was initiated by supervisory team in 2010 and rebuild was initiated in 2012 when the study began to propose a functional dual axis construct simulator capable of performing concurrent tasks in a predefined timeframe.
This experimental device is fully controlled by a multi-threaded software designed and implemented as a Load Cell Measurement Pilot (LCM Pilot).

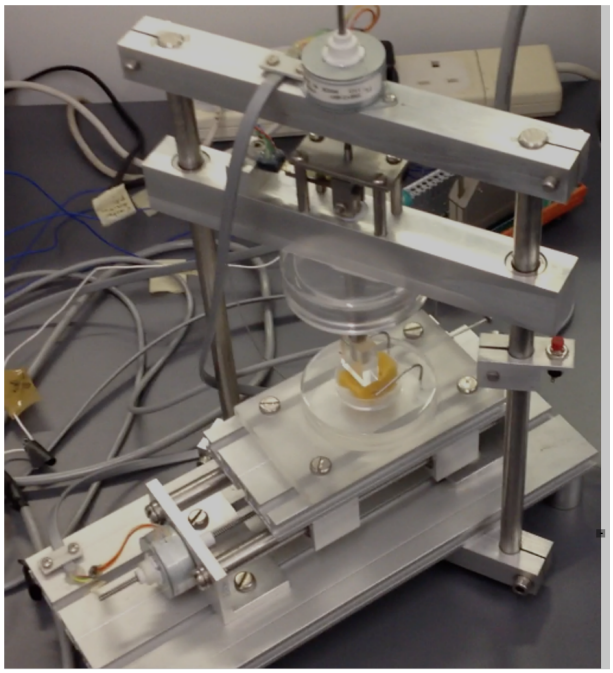

Figure 1 – Dual axis simulator to simulate active motion of knee joint.

LCM Pilot at its first stable release is a console based desktop application that is both automated and interactive. To initiate an experiment, LCM Pilot will collect author name and experiment settings. To configure an experiment following information will be collected as experiment setting from user:

- Number of sessions
- Load cycle in each session
- Load unit (in grams)
- Shear depth
- Resting period (in hours)
- Media cycle in a specified minute(s) interval

A brief description about the setting is provided below. Copy of source code is also available in appendix section, Apparatus code-base.

Experiment Settings

Experimental evidences suggest that the biosynthetic activity of CHs is regulated by the mechanical environment and it is time dependent (Wu et al., 1999). It is also reported that repetitive loading of AC at physiological levels of stress (1 MPa) is only harmful to CH in superficial zone and dependent on characteristics of load (static or cyclic) and duration of applied load (1-72h) (Lucchinetti et al., 2002).

To facilitate a mechanical environment for CHs under study, an experiment can be configured to run in number of sessions. In a multi-session setup, each session is isolated by a resting period defined in hours. The sessions also share load unit, load cycle and shear depth. This system is ideal to run experiments that can last for month. The term isolated sessions is used as strain data collected from each session are automatically uploaded to a remote location for online data chart plotting (Figure 2).

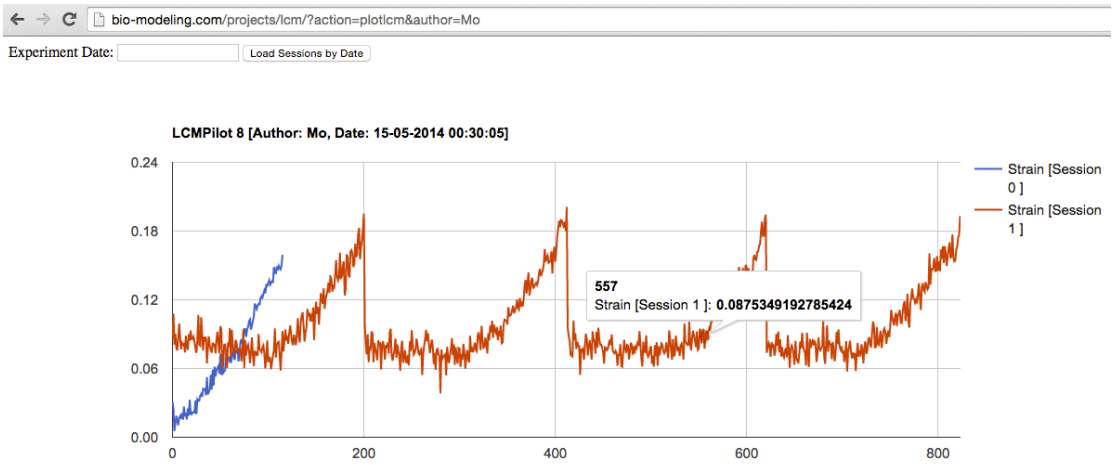

Figure 2 – Online displacement against strain data analysis for a multi-session experiments on wet/dry sponge

The result of each experiment will be stored remotely and the author can inspect the results by author name and experiment date. The system can save the data in CSV format or upload the data onto a remote server to visualise the data. This feature can be explored in page 43 in Appendix section. The format that the data will be available to a third party server will be in JSON format. This can be used to demonstrate strain changes during cyclic load in each session. This service can be extended and made available to other universities for collaboration and remote run of experimental studies and will also help to collect additional data for this study.

Managed Multi-threading

To make sure that each session runs smoothly with the provided setting, sequence of events must take place in specific order. The most popular mechanism for this is to implement concurrent software with threads. A threaded facility in a multi-threaded computer environment allows implementation of LCM Pilot software with multiple simultaneous points of execution, synchronising through shared memory (Birrell, 2003).

Multi-threading comes with set of challenges, to reduce complexity one can queue all tasks to be executed by thread pool threads. However, this study needs to address a rather more complicated situation where multiple threads require coordinated work. The final solution should be able to handle threads that block. This is to address the deadlocks and race conditions.

A deadlock happens when two threads tries to lock a resource that has already been locked by another. In this case none of the two threads can make any progress. Microsoft recommends use of time-outs to help detect deadlocks, however this introduces problems in an application that should run series of task without human intervention. In this case, flags are used to check the state of resources and only will obtain lock based on desired state of resources. At each cycle of an event such as winding a vertical motor or winding back a vertical motor the state of vertical motor and other horizontal motor is closely monitored.

Once lock is obtained, and enters a monitor the task gets executed and upon completion of the task monitor sends a pulse to notify all waiting threads of a change in the object state. This ensures a coordinated work of threads.
Another issue that must be tackled is the race conditions. A race condition can occur when the result of an application depends on which one or more threads reaches a section of code first. In our case this happens when the pumping task gets initiated in a timer thread. Given that this task should only take place at the end of each cycle and at an interval configured by the researcher, it is crucially important to only allow the task to take place when all other resources are at resting period and ensure that the samples under test are kept in living condition. Microsoft recommends a technique called synchronising data for multithreading. But race conditions can also occur when activities of multiple threads are synchronised. To overcome this, we have put the media pump task in a timer thread, this task will only take place if the state of other resources is at resting.

Cyclic Load

In a single load cycle, the perpendicular or axial force is applied during indentation, while the load riches its maximum unit the shearing force is applied to meet a predefined shear depth. Only then, in turn the axial and shear forces will be released. During each experiment, it is important to keep the cells viability intact, therefore a mechanism is designed and controlled by LCMPilot to provide media feed during resting period between each session. A two-way flow media pump system is custom built.
Two micro range of peristaltic pumps (model 100.005.012.030/4) with 4 rollers were utilised to control the media flow in and out of petri dish (Figure 4). The 12v DC powered motors have a flow rate of 1 ml per minute and require silicon tube for optimum performance. One end of tube connects to the media source and the other end is free. This isolation helps the liquid in the tube to remain sterile (M Y Jaffrin, 1971). These motors are controlled by LCMPilot via two 5V Single Pole DIL Reed Relays (Figure 3).

While calibrating the pump, to reach the desired flow the tube must be filled with the required media. That means the liquid must fill the entire tube and reach the free end. Later the free ends of the two tubes will be attached to dedicated micro pipelines that control the media flow in the dish.

Figure 4 – Peristaltic pumps, dc powered with 4 rollers

Figure 3 - 5v Single Pole DIL Reed Relay

The exact turn of events in each cycle is highlighted in an activity diagram (*Figure 5*)

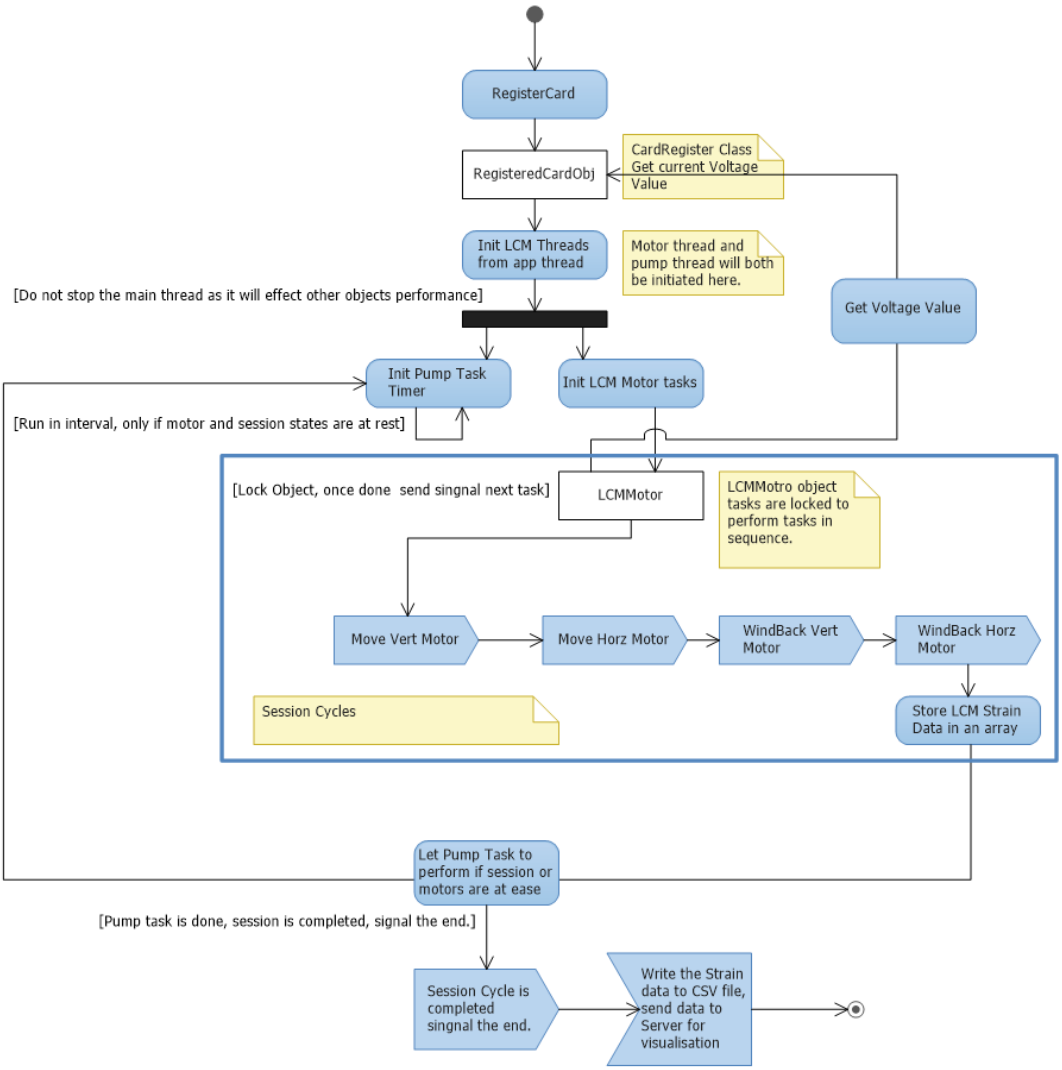

Figure 5 - LCM Pilot Activity diagram, a multi-threaded approach to control a custom build apparatus to mimic the in vivo loading regime of Articular Cartilage.

Unfortunately use of threads introduces various problems such as all tasks will compete for available resource (Lewis, 1995). In the LCMPilot setup all tasks must take place in specific order. We have used a Lock, Monitor, wait, Pulse and conventional Flag technique in Microsoft .Net Framework to ensure that execution of tasks are kept in order. The flag is a Boolean object that evaluates the state of other task in a Lock statement.

Lock is a primitive tool that offers mutual exclusion (also known as critical section), specifying for a particular region of code that only one thread can execute at any time. The monitor class implements wait and pulse methods to control the flow of events in a threaded application (Birrell, 2003). With the help of this technique we have managed to ensure that the media pump will only work when both axial and shearing forces are relieved and that shearing stress is only applied when the axial loading meets the specified load unit defined in an experiment. Moreover it also ensures that defined resting periods (in hours) and media pumps (in minutes) take effect with a good degree of accuracy.

To better understand how these tasks are controlled by LCMPilot, schematics of the custom design testing apparatus are provided in Figure 6 and Figure 7. Important components are numbered and further description will be provided accordingly.

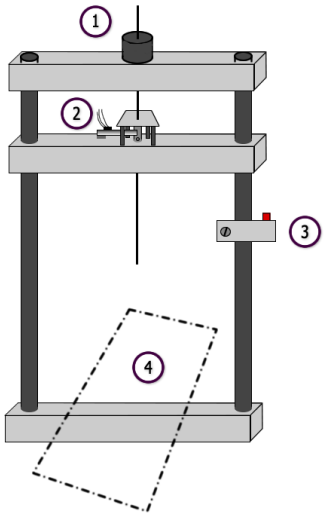

Figure 7 - Schematic of the custom designed apparatus

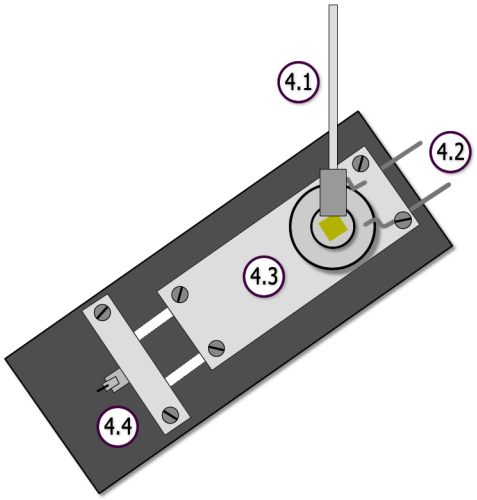

Figure 6 - second component of custom build apparatus, required for shearing stress

Component 1 and 4.4

There are two Hybrid 5v stepping motors that are in charge of perpendicular and horizontal movement. There are three types of stepping motors, variable reluctant, permanent magnet and hybrid. The developed LCM machine utilises hybrid stepping motors. These stepping motors are pulsed through a PCI 230 multi-functional analog and digital input/output amplicon board. The sequence and speed of applied pulses is directly related to motors' shaft rotation (direction and speed).
LCMPilot uses Amplicon component and micro motion controllers to change direction or speed.
This is achieved by sending TTL signals through analog channels in the PCI board. Exact channel numbers are highlighted in the code-base section in appendix.

Component 2

A strain gauge load cell LQB 630 (Thin Film Load Cell) with load range up to 50g is used to measure the applied axial force (Figure 8). LCMPilot requires constant feedback from this component. A load cell is a transducer that is used to create electrical signal that its magnitude is directly proportional to the force being applied (Gupta, 2012). In another word the mass is measured by converting the measured quantity into an electrical output and vice-versa. There are number of load cells such as hydraulic, pneumatic and strain gauge. The strain gauge load cell is utilised to measure the strain feedback in the apparatus.

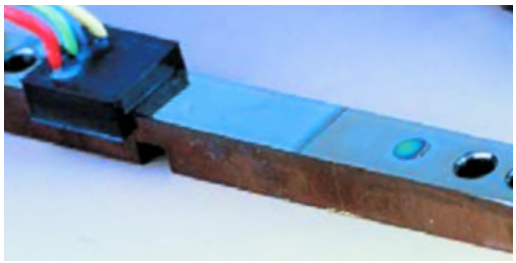

Figure 8 - Strain Gauge Load Cell

One end of this load cell is secured to a raised base and the other end is secured onto the ACME Screw perpendicular to the petri dish. The feedback is analysed to measure resistance response and also ensure that load unit is accurately applied.

Signal Amplification

The output of strain gauge is relatively small. According to the manufacturer of the strain gauge used in the apparatus the output is 1mv/V (1mV of output per volt excitation voltage). With the 10V excitation voltage the output signal will be 10mV. Therefore a signal amplifier is used to boost the signal level to increase measurement resolution and improve signal to noise ratio. The strain gauge amplifier is a Mantracourt SGA/D signal conditioner with gains up to 30.30mV. This amplifier has a dedicated switch to set variable gain in experiments.
It is very unlikely that the gauge will output exactly zero volts when no strain is applied. This is also referred to as initial offset voltage. There are few ways that a system can handle this initial offset voltage. The most common techniques recommended by manufacturers are software compensation, offset-nulling circuit, buffered offset-nulling and shunt calibration. SGA/D offers Zero Offset switch that can be used to compensate for the transducer zero error (the noise ratio).

Component 3

Physical safety button to halt the system in case the motor angular displacement cause the indentor to penetrate the sample in the petri dish and go through the dish as well. There is a software halt mechanism in place, but may malfunction at some point. Hence the need for a physical reset button in unattended experiments.

Component 4

Secures the petri dish mounted on and also applies shear stress during each cycle.

Component 4.1

The plane ended indentor with possibility to mount indenters of different cross sections such as round, square or rectangular shapes.

Component 4.2

The two media tubes are designed to refresh the media content in the petri dish. 2ml of media needs to be delivered during media replacement time in each session.
Component 4.3

The petri dish, with 5 ml working volume.

4. Testing

In this section the test strategy implementation is discussed as a proactive approach that helped to find and fix the defects before the final build. Throughout this process the functional behaviour of each requirement was tested using the black box software testing method. This is a method that requires no knowledge of the internal structure of LCMPilot and its internal structure, design and implementation.

Pump Calibration

Media pumps were primed for testing and timing adjustments in the software.
The tubes that are connecting the media source to petri dish were disconnected to be prefilled with water. Distilled water was injected from one end of tube until filled the entire tube. This was repeated for second tube, as two tubes are required to control the media flow in and out of petri dish.

Disconnected tubes were reconnected and pump motors were energised in turn to assess the flow rate in each tube. Average flow rate was assessed at 0.8ml per minute which is closely similar to manufacturer specification. On average it takes 5 minutes to deliver 4/ml media from source to petri dish and 5 minutes to take 3.65ml to draw media out of petri dish. Petri dish overall volume is 5ml and weights 1.51g.

Soft Testing

In soft testing stage, the LCMPilot software and DACS hardware were tested against series of experiments to evaluate software functionality and to assess the accuracy of tasks execution. Further the tests revealed several hardware faults; as a result, two hardware components required replacement.

In an isolated pump calibration session, in series of setup configurations to prime the pump, the two relay motors controlling the pump were malfunctioning and a physical tap on relays was required to energise them. In a circuit current investigation it revealed that the two relays were drawing too much current from the PCI board (10V from a 5V base output). Therefore the malfunctioned relays were replaced by a new pair as mentioned above.
The second component that needed to be changed was the strain gauge with load range up to 100g. The strain gauge was functioning during soft testing period in experiments where a wet/dry sponge were used as the load bearing material. During these testing sessions, overall functionality of DACS and pumping mechanism was tested under variable loads up to 45g loading regimes. Collected results during testing or chart plotting in figures are based on this setup.

This is a considerable load (45g) when it comes to the culture system that was previously discussed and will damage the gel. In first stage of hard testing, the first few cycles fractured the several plate of agarose gels that was casted for series of experiments. This insensitivity to load resistance was the result of strain gauge malfunction. In an isolated calibration session the strain gauge signals were examined. In conclusion the strain gauge had to be replaced by a new model with load range up to 50g.

Hard Testing

A hard testing plan was organised to test the apparatus functionality in vitro. Artificial matrix systems such as agarose gel was used as the matrix properties are similar to those of cell (Freeman et al., 1994, Knight et al., 1998).

Several gels with different consistency were prepared for testing.
0.4 grams of agarose was mixed with 40ml distilled water used as buffer and placed in microwave and heat (one minute interval) until agarose was dissolved and boiling. While the mix was cold enough to handle but not set, it was poured into the petri dish with 5 mil capacity. In about 10 minutes the agarose harden and it changed colour from clear to slightly opaque. This has produced ~1% gel consistency. This is towards a quite hard consistency we needed as a sample. Other gels where casted with agarose to water ratio of 0.06g agarose/50ml water and 0.006g agarose to 60ml water.

These gels were stored in fridge until needed for duration of testing.

The first series of testing was conducted on agarose gel with different consistency (from hard to soft) to test the strain signals. During testing the gels were damaged considerably. As previously reported, the strain gauge had to be replaced to fix the issue.

Further tests need to be carried out in vitro, within an incubator. DACS must be connected to a PC and the PC must be connected to internet at all times. Long cables are used in developing the connections to facilitate the portability required for storing the DACS in an incubator.

5 Conclusion

The major focus of this study so far was to design and develop the custom built apparatus and the multi-threaded LCM Pilot software. In addition extensive literature review was required to understand the bio-mechanical concepts of this study. This would be followed through further with initial molecular biology and cell culture training. Further literature review is required to better gain knowledge about internal bio-mechanism of cartilage matrix. Also further studies are required to learn about gel casting techniques, maintenance and its storage. Lab training will need to take place to conduct live and hard testing.
The training will need to be provided by supervisory team to gain skills and knowledge about various techniques required to culture and run experiments to study chondrocytes.

There are standard series of testing and analytical techniques that can be performed to compare and study results that will be produced in the experiments. Some of which can be named as follow: Biomechanical creep curve testing, paravital staining used for microscopic assessment of chondrocytes viability (Hurtig et al., 1998, Clements et al., 2001), histological analysis, and gross morphological analysis.
As this is a multi-disciplinary study a combined knowledge of both computer and life science are crucial. The development of the DACS and its multi-threaded application plays important factor in this experimental study of loaded articular cartilage in cellular level. Series of in vitro experiments would be planned to work on the effect of fluid movement within the cartilage matrix and also the shape of the intra-articular contact area within joints and its effect on joint health.

It is important to note that studies in existing literature as highlighted in introduction are subject to assumptions made regarding the constitutive behaviour, geometry, specimen type/maturity, loading regime and in vitro conditions of the conducted experiments. Idealising problems has previously helped scientists to present modelling techniques for more linear problems.

5.1 Future Work

Future work will involve learning about gel casting, staining techniques, maintenance and storage. Different staining techniques and microscopic assessments will be used to assess chondrocyte viability under study.

There are number of standard statistical analysis, and several techniques would be used to study and evaluate the experiments' results. Results will be presented as mean standard error of the human (SEM). Variation in cell death with zone dependency will be examined using one-way analysis of variance (ANOVA). Results can also be analysed using Tukey's multiple comparison test, as well as F-test and paired or unpaired t-test. Significance will be analysed at $p < 0.05$. These statistical analysis can be performed with Matlab, Microsoft Excel or Graphpad Prism.

As part of experiments development, C-20/A4 cells will be cultured in type I collagen gel and studied under compression with interchangeable indentor in a custom built apparatus called DACS. The culture system will be studied under dynamic load first, followed by dual axis loading in vitro.

Future works beyond this research could also involve 3 dimensional finite element modelling of bio-mechanical systems and evaluation of existing techniques to propose a hybrid approach on modelling the bio-mechanical behaviour of articular cartilage in cellular level. This study could be extended to develop and work on type II collagen gel culture system for the culture of immortalised and primary chondrocytes. A comparison study could also be conducted in the use of different type of chondrocytes cell lines such as C-28/I2, T/C-28a2 and T/C28a4 cell lines.

6 Appendix

6.1 Apparatus Code-Base

The main program file. This file will be executed first.
Program.cs

```
using System;
using System.ComponentModel;
using System.Data;
using System.Collections;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using LCMConsole.Entities;
using LCMConsole.Utilities;
using System.Runtime.InteropServices;
using System.Runtime.CompilerServices;
using Amplicon.AmpDIO;
using System.IO;

namespace LCMConsole
{
    public class Program
    {
        static void Main(string[] args)
        {

            int result = 0;   // Result initialized to say
there is no error
            // sessions cycles
            int sessionCycles = 0;
            // session load cycles
            int sessionLoadCycles = 0;
            //session sleeping period
            int sessionSleepingPeriodHr = 0;
            // media pump cycle
            int mediaPumpCycleMin = 0;
            //required load
            double requiredLoad = 0.0;
            //sheer depth
            int sheerDepth = 0;

            //register board
```

```
            RegisterCard registerBoardObj = new
RegisterCard();
            registerBoardObj.registerMyBoard();

            Console.WriteLine("Enter number of sessions:");
            int.TryParse(Console.ReadLine(), out
sessionCycles);
            //load cycle per session
            Console.WriteLine("Enter number of sessions Load
Cycles:");
            int.TryParse(Console.ReadLine(), out
sessionLoadCycles);
            //required load
            Console.WriteLine("Enter the required load:");
            double.TryParse(Console.ReadLine(), out
requiredLoad);
            //sheer depth
            Console.WriteLine("Enter Sheer Depth:");
            int.TryParse(Console.ReadLine(), out sheerDepth);
            //init the session resting period
            Console.WriteLine("Enter Session Resting Period
(in hours):");
            int.TryParse(Console.ReadLine(), out
sessionSleepingPeriodHr);

            //init the lcm motor
            Console.WriteLine("Initialising the LCM Motors
...");
            LCMMotor lcmMotor = new LCMMotor(sessionCycles,
sessionLoadCycles);
            lcmMotor.RequiredLoad = requiredLoad;
            lcmMotor.SheerDepth = sheerDepth;
            lcmMotor.RegisterBoardObj = registerBoardObj;
            lcmMotor.SessionSleepHr = sessionSleepingPeriodHr;

            //init the media pump cycle
            Console.WriteLine("Enter Media Pump cycle (every X
minutes):");
            int.TryParse(Console.ReadLine(), out
mediaPumpCycleMin);
            //Media pump cycle in min
            lcmMotor.PumpCycleMin = mediaPumpCycleMin;

            //get the author name
            Console.WriteLine("Enter the author name:");
            lcmMotor.AuhtorName = Console.ReadLine();

            //get the Meterdata
            lcmMotor.MeterData = registerBoardObj.MeterData;
```

```
            //thread timer delegate to call the analogue task
            System.Threading.TimerCallback
AnalogueTimerDelegate = new
System.Threading.TimerCallback(lcmMotor.getAnalogueTask);

            System.Threading.Timer analogueTimerItem = new
System.Threading.Timer(AnalogueTimerDelegate, lcmMotor, 0,
250);

            lcmMotor.GetAnalogueReference = analogueTimerItem;

            //start LCM motor consumer
            Thread motorConsumer = new Thread(new
ThreadStart(lcmMotor.threadRun));



            long pumpFirstTriggerTime =
(long)lcmMotor.pumpIntervalMili();
            long pumpTriggerTimeInterval =
(long)lcmMotor.pumpIntervalMili();


            //thread timer delegate to call the pump task
            System.Threading.TimerCallback TimerDelegate = new
System.Threading.TimerCallback(lcmMotor.pumpTimerTask);

            // Timer calls the media pump task.
            // the timer starts running as soon as the
instance is created.
            System.Threading.Timer TimerItem = new
System.Threading.Timer(TimerDelegate, lcmMotor,
pumpFirstTriggerTime, pumpTriggerTimeInterval);

            lcmMotor.PumpTimerReference = TimerItem;


            //start the thrads and catch exceptions
            try
            {
                motorConsumer.Start();
                //Join threads with no timeout
                // Run until done.
                motorConsumer.Join();
                // threads producer and consumer have finished
at this point.
            }
            catch (ThreadStateException e)
```

```
            {
                Console.WriteLine(e);  // Display text of
exception
                result = 1;            // Result says there
was an error
            }
            catch (ThreadInterruptedException e)
            {
                Console.WriteLine(e);  // This exception means
that the thread
                // was interrupted during a Wait
                result = 1;            // Result says there
was an error
            }
            // Even though Main returns void, this provides a
return code to
            // the parent process.
            Environment.ExitCode = result;
        }


    }
}
```

Control motors motions, media flow and sequence of events/tasks.

LCMMotor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using LCMConsole.Utilities;
using LCMConsole.Entities;
using Amplicon.AmpDIO;
using System.IO;
using System.Text;
using System.Data;

namespace LCMConsole.Entities
{
    class LCMMotor
    {
        //Stepper motors related properties and methods
        private bool isVertMotorRunning = false;

        public bool IsVertMotorRunning
        {
            get { return isVertMotorRunning; }
```

```
        set { isVertMotorRunning = value; }
    }
    private bool isHorMotorRunning = false;

    public bool IsHorMotorRunning
    {
        get { return isHorMotorRunning; }
        set { isHorMotorRunning = value; }
    }

    //In calibrated position state is true
    private bool vertMotorState = true;

    public bool VertMotorState
    {
        get { return vertMotorState; }
        set { vertMotorState = value; }
    }

    //In calibrated position state is true
    private bool horMotorState = true;

    public bool HorMotorState
    {
        get { return horMotorState; }
        set { horMotorState = value; }
    }

    private int loadCell = 0;

    public int LoadCell
    {
        get { return loadCell; }
        set { loadCell = value; }
    }

    private int loadCycles = 0;

    public int LoadCycles
    {
        get { return loadCycles; }
        set { loadCycles = value; }
    }

    private bool stepperMotorSwitched = false;

    public bool StepperMotorSwitched
    {
        get { return stepperMotorSwitched; }
```

```
            set { stepperMotorSwitched = value; }
        }

        // Session related properties and methods
        private int sessionCycles = 0;

        public int SessionCycles
        {
            get { return sessionCycles; }
            set { sessionCycles = value; }
        }

        private bool sessionState = true;

        public bool SessionState
        {
            get { return sessionState; }
            set { sessionState = value; }
        }

        private double sessionSleepHr;

        public double SessionSleepHr
        {
            get { return sessionSleepHr; }
            set { sessionSleepHr = value; }
        }

        private TimeSpan sessionSleepMili;

        public TimeSpan SessionSleepMili
        {
            get { return sessionSleepMili; }
            set { sessionSleepMili = value; }
        }

        public LCMMotor(int sessionCycles = 0, int
sessionLoadCycle = 0)
        {
            SessionCycles = sessionCycles;
            LoadCycles = sessionLoadCycle;
        }

        //Media pump related properties and methods

        //Media pump is not running and not in use, so
following properties are set to false
        private bool isPumpRunning = false;
        private bool pumpState = false;
```

```
public bool PumpState
{
    get { return pumpState; }
    set { pumpState = value; }
}

public bool IsPumpRunning
{
    get { return isPumpRunning; }
    set { isPumpRunning = value; }
}
private double pumpCycleMin = 0;

public double PumpCycleMin
{
    get { return pumpCycleMin; }
    set { pumpCycleMin = value; }
}

private System.Threading.Timer pumpTimerReference;

public System.Threading.Timer PumpTimerReference
{
    get { return pumpTimerReference; }
    set { pumpTimerReference = value; }
}

private System.Threading.Timer getAnalogueReference;

public System.Threading.Timer GetAnalogueReference
{
    get { return getAnalogueReference; }
    set { getAnalogueReference = value; }
}

//Get current voltage, load and MeterData
private double meterData;

public double MeterData
{
    get { return meterData; }
    set { meterData = value; }
}

private double currentLoadValue = 0.0;

double requiredLoad;
```

```
        public double RequiredLoad
        {
            get
            {
                //convert the load to map the calibration
                //so this means 0.64 is 100% percent
(normalised)
                return requiredLoad / 156.0;
            }
            set { requiredLoad = value; }
        }

        //sheer depth in unites
        private int sheerDepth = 0;

        public int SheerDepth
        {
            get { return sheerDepth; }
            set { sheerDepth = value; }
        }

        //current sheer depth
        private int currentSheerDepth = 0;

        public int CurrentSheerDepth
        {
            get { return currentSheerDepth; }
            set { currentSheerDepth = value; }
        }

        private RegisterCard registerBoardObj;

        internal RegisterCard RegisterBoardObj
        {
            get { return registerBoardObj; }
            set { registerBoardObj = value; }
        }

        //get/set author name
        private string auhtorName;
        public string AuhtorName
        {
            get { return auhtorName; }
            set { auhtorName = value; }
        }



        //act as producer
```

```
        public void runVertMotor(int vcounter)
        {
            lock (this)
            {
                if (StepperMotorSwitched && PumpState)
                {
                    Monitor.Wait(this);
                }

                Console.WriteLine("Vert Motor cycle @" +
vcounter);
                //change motor direction, going down
                DIO_TC.DIOsetData(0, 0, 1, 1);

                //need to get the load cell data in here
                //get the load value and apply the calibration
                currentLoadValue =
RegisterBoardObj.CurrentLoadValue;

                while (currentLoadValue < RequiredLoad)//this
is the load cell feedback
                {
                    RegisterBoardObj.getAnalogueValue();
                    DIO_TC.DIOsetData(0, 0, 0, 1);
                    //creates the pulse, adjust the the speed
                    Thread.Sleep(100);
                    DIO_TC.DIOsetData(0, 0, 0, 0);
                    MotorState.VertMotorPos++;
                    currentLoadValue =
RegisterBoardObj.CurrentLoadValue;
                    //prepare the data array to write into a
csv file
                    prepDataArray();
                }

                StepperMotorSwitched = true;
                VertMotorState = false;
                IsVertMotorRunning = true;
                Monitor.PulseAll(this);

            }

        }

        //act as consumer
        public void runHorMotor(int hcounter)
        {
            lock (this)
            {
```

```
                if (!StepperMotorSwitched && PumpState)
                {
                    Monitor.Wait(this);
                }

                Console.WriteLine("Horiz Motor cycle @" +
hcounter);
                //change the motor direction, going in
                DIO_TC.DIOsetData(0, 0, 3, 1);
                CurrentSheerDepth = 0;
                while (CurrentSheerDepth < SheerDepth)
                {
                    //move the motor to the required shear
depth
                    //set channel A2 high - pin 14
                    DIO_TC.DIOsetData(0, 0, 2, 1);
                    //adjust speed and pulse
                    Thread.Sleep(100);
                    //set channel A2 low - pin 14
                    DIO_TC.DIOsetData(0, 0, 2, 0);
                    CurrentSheerDepth++;
                    MotorState.HorMotorPos++;
                }

                HorMotorState = false;
                StepperMotorSwitched = false;
                IsHorMotorRunning = true;
                Monitor.PulseAll(this);
            }

        }

        public void windBackVert(int wvcounter)
        {
            lock (this)
            {
                if (!IsVertMotorRunning && PumpState)
                {
                    Monitor.Wait(this);
                }

                Console.WriteLine("WindBack Vert Motor @" +
wvcounter);

                //reverse the motor direction
                DIO_TC.DIOsetData(0, 0, 1, 0);
                MotorState.VertMotorGoingUp = true;
                //preserve the original position
                int motorPos = MotorState.VertMotorPos;
```

```
                while (MotorState.VertMotorPos > 0)
                {
                    DIO_TC.DIOsetData(0, 0, 0, 1);
                    //adjust speed and pulse
                    Thread.Sleep(100);
                    DIO_TC.DIOsetData(0, 0, 0, 0);
                    MotorState.VertMotorPos--;
                }

                VertMotorState = true;
                IsVertMotorRunning = false;
                Monitor.PulseAll(this);
            }
        }

        public void windBackHor(int whcounter)
        {
            lock (this)
            {
                if (!IsHorMotorRunning && PumpState)
                {
                    Monitor.Wait(this);
                }

                Console.WriteLine("WindBack Hor Motor @" +
whcounter);

                //reverse the motor direction
                DIO_TC.DIOsetData(0, 0, 3, 0);
                MotorState.HorMotorGoingOut = true;
                while (MotorState.HorMotorPos > 0)
                {
                    //set channel A2 high - pin 14
                    DIO_TC.DIOsetData(0, 0, 2, 1);
                    //adjust speed and pulse
                    Thread.Sleep(100);
                    //set channel A2 low - pin 14
                    DIO_TC.DIOsetData(0, 0, 2, 0);
                    MotorState.HorMotorPos--;
                }

                HorMotorState = true;
                IsHorMotorRunning = false;
                Monitor.PulseAll(this);
            }
        }

        public void runSession(int counter)
        {
```

```
            lock (this)
            {
                if (!SessionState && PumpState)
                {
                    Monitor.Wait(this);
                }

                DateTime now = DateTime.Now;
                Console.WriteLine("**************Session
Started @ " + counter + " @ " + now + "****************");
                SessionState = false;
                Monitor.PulseAll(this);
            }
        }

        public void runMediaPump()
        {

            if (PumpState && !IsVertMotorRunning &&
!IsHorMotorRunning)
            {
                DateTime now = DateTime.Now;
                Console.WriteLine("**************Media pump is
started @ " + now + "****************");
                //pump is running, 1ml per 1min make sure no
other tasks are running
                IsPumpRunning = true;

                //flush the medium
                //first switch on pump A - out - pin 13
                DIO_TC.DIOsetData(0, 0, 4, 1);
                //wait one second
                Thread.Sleep(60000);
                DIO_TC.DIOsetData(0, 0, 4, 0);
                //then switch on pump B - in - pin 12
                DIO_TC.DIOsetData(0, 0, 6, 1);
                //wait for some time
                Thread.Sleep(60000);
                //switch off pump B
                DIO_TC.DIOsetData(0, 0, 6, 0);

                now = DateTime.Now;
                Console.WriteLine("**************Media pump is
stopped @ " + now + "****************");
                IsPumpRunning = false;
                PumpState = false;
            }
        }
```

```
        public void threadRun()
        {
            for (int counter = 1; counter <= SessionCycles;
counter++)
            {
                runSession(counter);
                for (int htcounter = 1; htcounter <=
LoadCycles; htcounter++)
                {
                    runVertMotor(htcounter);
                    runHorMotor(htcounter);
                    windBackVert(htcounter);
                    windBackHor(htcounter);
                    //sleeping time between each cycle
                    Task.Delay(1000).Wait();
                }

                GetAnalogueReference.Dispose();

                if (counter != SessionCycles)
                {
                    sessionSleepingTimer();
                }
            }
            //dispose the pump timer function
            if (!PumpState)
            {
                PumpTimerReference.Dispose();
            }

            //at the end, write to the file and also send the
file to server for charts
            FileWriter fw = new FileWriter("LCM_" +
DateTime.Now.ToString("yyyyMMddHHmm"));
            fw.WriteData(vmotorPos, dataValue,
dataValue.Length);

            //send data to server
            ClientApp ca = new ClientApp();
            ca.sendData(fw.FileName, AuhtorName);

        }

        public void sessionSleepingTimer()
        {
            if (SessionState == false)
            {
```

```
                Console.WriteLine("************** Session in
sleeping mode, resume in " + SessionSleepHr +
"hrs****************");
                //hour to millisecond
                //SessionSleepMili =
TimeSpan.FromMilliseconds(TimeSpan.FromHours(SessionSleepHr).T
otalMilliseconds);
                SessionSleepMili =
TimeSpan.FromMilliseconds(TimeSpan.FromMinutes(SessionSleepHr)
.TotalMilliseconds);
                Task.Delay(SessionSleepMili).Wait();
                SessionState = true;
                //Monitor.Pulse(this);
            }
        }

        public double pumpIntervalMili()
        {
            return
TimeSpan.FromMinutes(PumpCycleMin).TotalMilliseconds;

        }

        public void pumpTimerTask(object stateobj)
        {
            lock (this)
            {
                if (stateobj is LCMMotor)
                {
                    LCMMotor lcmobj = (LCMMotor)stateobj;
                    PumpState = true;
                    runMediaPump();
                }
            }


        }

        public void getAnalogueTask(object stateobj)
        {
            lock (this)
            {
                if (stateobj is LCMMotor)
                {
                    LCMMotor lcmobj = (LCMMotor)stateobj;
                    RegisterBoardObj.getAnalogueValue();
                }
            }
        }
```

```
        #region FileWriter
        double[] dataValue = new double[10000];
        //use an ArrayList here
        public double[] DataValue
        {
            get { return dataValue; }
            set { dataValue = value; }
        }
        //use an ArrayList here
        int[] vmotorPos = new int[10000];

        public int[] VmotorPos
        {
            get { return vmotorPos; }
            set { vmotorPos = value; }s
        }

        int fileWriteCounter = 0;

        public int FileWriteCounter
        {
            get { return fileWriteCounter; }
            set { fileWriteCounter = value; }
        }

        private void prepDataArray()
        {
            //update the file with the voltage/load
            //write the file at the end of the cycle
            //only write to file on depression
            //write vertical position
            if (!MotorState.LoadReached)
            {
                //vmotorPos[fileWriteCounter] =
MotorState.VertMotorPos;
                vmotorPos[fileWriteCounter] =
FileWriteCounter;
                dataValue[fileWriteCounter] =
Calibration.getNewtons(RegisterBoardObj.MeterData);
                FileWriteCounter++;
            }


        }
        #endregion
    }
}
```

LCMSession.cs Control session, and read/get session state

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace LCMConsole.Entities
{
    class LCMSession
    {
        private bool sessionState = true;

        public bool SessionState
        {
            get { return sessionState; }
            set { sessionState = value; }
        }
        private bool sessionIsSleeping = false;

        public bool SessionIsSleeping
        {
            get { return sessionIsSleeping; }
            set { sessionIsSleeping = value; }
        }
        private int sessionCycles = 0;

        public int SessionCycles
        {
            get { return sessionCycles; }
            set { sessionCycles = value; }
        }
        private int sessionSleepHr = 8;

        public int SessionSleepHr
        {
            get { return sessionSleepHr; }
            set { sessionSleepHr = value; }
        }
        private int sessionLoadCycle = 30;
```

```
        public int SessionLoadCycle
        {
            get { return sessionLoadCycle; }
            set { sessionLoadCycle = value; }
        }
        private bool sessionIsRunning = false;

        public bool SessionIsRunning
        {
            get { return sessionIsRunning; }
            set { sessionIsRunning = value; }
        }

        private int sessionLoadCycleSleepSec = 0;

        public int SessionLoadCycleSleepSec
        {
            get { return sessionLoadCycleSleepSec; }
            set { sessionLoadCycleSleepSec = value; }
        }

        public LCMSession(int _sessionCycles = 0, int
_sessionLoadCycle = 0)
        {
            SessionCycles = _sessionCycles;
            SessionLoadCycle = _sessionLoadCycle;
        }

        public void runSession(int counter)
        {
            lock (this)
            {
                if (!SessionState)
                {
                    Monitor.Wait(this);
                }

                Console.WriteLine("**************Session
Started @" + counter);
                SessionState = false;
                Monitor.Pulse(this);
            }
        }

        public void writeSessionData()
        {

        }
```

```
        public void writeSessionDataThread()
        {

        }

        public void threadRun()
        {
            for (int counter = 1; counter <= SessionCycles;
counter++)
            {
                runSession(counter);
            }
        }
    }
}
```

Control flow of media, and task state.

LCMMedia.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LCMConsole.Entities
{
    class LCMMedia
    {
        private bool isPumpRunning = false;
        private bool pumpState = false;

        public bool PompState
        {
            get { return pumpState; }
            set { pumpState = value; }
        }

        public bool IsPumpRunning
        {
            get { return isPumpRunning; }
            set { isPumpRunning = value; }
        }
        private int pumpCycleMin = 0;

        public int PumpCycleMin
        {
            get { return pumpCycleMin; }
```

```
            set { pumpCycleMin = value; }
        }

        public void runPump()
        {

        }

        public void ThreadRun()
        {

        }
    }
}
```

Read and get State of motors at one time

MotorState.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LCMConsole.Entities
{
    static class MotorState
    {
        static Boolean horzMotorCalibrated = false;

        public static Boolean HorzMotorCalibrated
        {
            get { return MotorState.horzMotorCalibrated; }
            set { MotorState.horzMotorCalibrated = value; }
        }
        static Boolean vertMotorCalibrated = false;

        public static Boolean VertMotorCalibrated
        {
            get { return MotorState.vertMotorCalibrated; }
            set { MotorState.vertMotorCalibrated = value; }
        }
        static Boolean hasCompletedCycle = false;

        public static Boolean HasCompletedCycle
        {
            get { return MotorState.hasCompletedCycle; }
            set { MotorState.hasCompletedCycle = value; }
        }
```

```
static Boolean loadReached;

public static Boolean LoadReached
{
    get { return MotorState.loadReached; }
    set { MotorState.loadReached = value; }
}

static Boolean verMotorEnabled = false;

public static Boolean VerMotorEnabled
{
    get { return MotorState.verMotorEnabled; }
    set { MotorState.verMotorEnabled = value; }
}
static Boolean horMotorEnabled = false;

public static Boolean HorMotorEnabled
{
    get { return MotorState.horMotorEnabled; }
    set { MotorState.horMotorEnabled = value; }
}
static int vertMotorPos = 0;

public static int VertMotorPos
{
    get { return MotorState.vertMotorPos; }
    set { MotorState.vertMotorPos = value; }
}
static int horMotorPos = 0;

public static int HorMotorPos
{
    get { return MotorState.horMotorPos; }
    set { MotorState.horMotorPos = value; }
}
static Boolean vertMotorStoppedonLoad = false;

public static Boolean VertMotorStoppedonLoad
{
    get { return MotorState.vertMotorStoppedonLoad; }
    set { MotorState.vertMotorStoppedonLoad = value; }
}
static Boolean vertMotorAtZero = false;

public static Boolean VertMotorAtZero
{
    get { return MotorState.vertMotorAtZero; }
    set { MotorState.vertMotorAtZero = value; }
```

```
        }
        static Boolean horMotorAtZero = false;

        public static Boolean HorMotorAtZero
        {
            get { return MotorState.horMotorAtZero; }
            set { MotorState.horMotorAtZero = value; }
        }
        private static Boolean horMotorGoingOut = false;

        public static Boolean HorMotorGoingOut
        {
            get { return MotorState.horMotorGoingOut; }
            set { MotorState.horMotorGoingOut = value; }
        }
        private static Boolean horMotorGoingIn = false;

        public static Boolean HorMotorGoingIn
        {
            get { return MotorState.horMotorGoingIn; }
            set { MotorState.horMotorGoingIn = value; }
        }
        private static Boolean vertMotorGoingUp = false;

        public static Boolean VertMotorGoingUp
        {
            get { return MotorState.vertMotorGoingUp; }
            set { MotorState.vertMotorGoingUp = value; }
        }
        private static Boolean vertMotorGoingDown = false;

        public static Boolean VertMotorGoingDown
        {
            get { return MotorState.vertMotorGoingDown; }
            set { MotorState.vertMotorGoingDown = value; }
        }
        private static Boolean horzMotorShearedSample;

        public static Boolean HorzMotorShearedSample
        {
            get { return MotorState.horzMotorShearedSample; }
            set { MotorState.horzMotorShearedSample = value; }
        }


    }
}
```

Load calibration, converting a load cell voltage to a force.

Calibration.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Numerics;
using System.Threading.Tasks;

namespace LCMConsole.Utilities
{
    /// <summary>
    /// This class is responsible for converting
    /// the load-cell voltage to a force.
    /// Static class.
    /// </summary>
    public static class Calibration
    {
        static double newtons;
        static double conversionFactor = 0;
        public static double getNewtons(double voltage)
        {
            //reaarange
            conversionFactor = 3.3512 * voltage + 27.714;
            newtons = conversionFactor / 100;
            return newtons;
        }
    }
}
```

Upload experiment result to a remote location for further analysis.

ClientApp.cs

```
using System;
using System.ServiceModel.Web;
using System.Runtime.Serialization.Json;
using System.Net;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LCMConsole.Utilities
{
    class ClientApp
    {
        //send LCM data to server

        public void sendData(string fileName, string auhtor)
        {
```

```
            String dir =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocument
s) + "\\GelLoaderData";
            String filePath = dir + "\\" + fileName;


            List<string[]> csv = new List<string[]>();

            var lines = System.IO.File.ReadAllLines(filePath);
            foreach (string line in lines)
                csv.Add(line.Split(','));

            //specifiy the url you want to send data to
            string serverURL = "[YOUR_REMOTE_SERVER_ADDRESS]"
+ auhtor;

            //make request to url and set post properties
            HttpWebRequest request =
(HttpWebRequest)WebRequest.Create(serverURL);
            request.Method = "POST";
            request.ContentType = "application/json;
charset=utf-8";

            try
            {
                //serialize
                DataContractJsonSerializer ser = new
DataContractJsonSerializer(typeof(List<string[]>));
                MemoryStream ms = new MemoryStream();
                ser.WriteObject(ms, csv);
                string lcmData =
Encoding.UTF8.GetString(ms.ToArray());

                StreamWriter writer = new
StreamWriter(request.GetRequestStream());
                writer.Write(lcmData);
                writer.Flush();
                writer.Close();

                HttpWebResponse response =
(HttpWebResponse)request.GetResponse();
                StreamReader streamReader = new
StreamReader(response.GetResponseStream());
                Console.WriteLine("LCM data is uploaded to the
server, response: " + streamReader.ReadToEnd());
                Console.WriteLine("You can review your data
chart at: http://bio-
modeling.com/projects/lcm/?action=plotlcm&author=" + auhtor);
```

```
            }
            catch (Exception ex)
            {
                Console.WriteLine("Unable to send data: " +
ex.Message);
            }
        }
    }
}
```

Save experiment result into a CSV file localy.

FileWriter.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;

namespace LCMConsole.Utilities
{
    class FileWriter
    {
        private String fileName;

        public String FileName
        {
            get { return fileName; }
            set { fileName = value; }
        }
        private FileStream file;

        public FileStream File
        {
            get { return file; }
            set { file = value; }
        }
        private StreamWriter sw;


        //constructor
        public FileWriter(string _filename)
        {
            FileName = _filename + ".csv";
            //create a directory if it doesn't exist
            String dir =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocument
s) + "\\GelLoaderData";
            if (!Directory.Exists(dir))
```

```
            {
                //Create it
                Directory.CreateDirectory(dir);

            }
            //ensure that we have uniques files
            if (!System.IO.File.Exists(dir + "/" + FileName))
            {
                using (System.IO.FileStream fs =
System.IO.File.Create(dir + "\\" + FileName))
                {
                    String completePath =
System.IO.Path.Combine(dir, FileName);
                    fs.Close();
                }

            }
            //the complete file path
            String filePath = dir + "\\" + FileName;
            //open the file
            File = new FileStream(filePath, FileMode.Open,
FileAccess.Write);
            sw = new StreamWriter(File);

        }

        public void WriteData(int[] motorPos, double[] force,
int numDataPoints)
        {
            for (int i = 0; i < numDataPoints; i++)
            {
                if (force[i] > 0.0)
                {
                    sw.WriteLine(motorPos[i].ToString() + ","
+ force[i].ToString());
                }
            }
            closeFile();
        }

        public void closeFile()
        {
            sw.Close();
            File.Close();

        }
    }
}
```

Read and write using PCI 230/260 component from Amplicon

RegisterCard.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Amplicon.AmpDIO;

namespace LCMConsole.Utilities
{
    public struct Board_Details_T
    {
        public short model;
        public short baseAddr;
        public short irq;
        public bool isPci;
        public short pciBus;
        public short pciSlot;
        public uint hwVersion;
        public string cardName;
        public string cardDesc;
    }

    class RegisterCard
    {
        private double UniScale = 4095.0 * 16;              //
Full scale unipolar
        private double BiScale = 2047.0 * 16;               //
Half scale bipolar
        private short G_hTCInt = -1;               // Handle
for TC Interrupts
        private short G_IntChip;              // Interrupt
event chip offset
        private int G_CurrentChannel = 0;       // Record of
Current Channel Selected
        private int G_CurrentPCI230_Range = 3;
        private bool G_Diff = false;
        private short G_SelectedCardHandle = -1;   // Handle
for the Card
        private Board_Details_T G_CardDetails;  // Details of
card
        private double G_VoltageScale = 1;
        private double loadCellData;
        private double meterData;
        private double currentLoadValue = 0.0;

        public double CurrentLoadValue
```

```
        {
            get { return currentLoadValue; }
            set { currentLoadValue = value; }
        }
        private DIO_TC.TTCCALLBACK cbProc;
        private Board_Details_T[] P_Boards = new
Board_Details_T[DIO_TC.NUMBER_CARD_SUPPORTED];
        public short hBoard;

        //getters and setters
        public double LoadCellData
        {
            get { return loadCellData; }
            set { loadCellData = value; }
        }
        public double MeterData
        {
            get { return meterData; }
            set { meterData = value; }
        }

        //-----------------------------------------------------
---------------------
        // GetBoardDetails
        //-----------------------------------------------------
---------------------
        // Function
        //   Fill in Detail Structure with information about
the board
        // Parameters
        //   The Handle for the Board
        // Returns
        //   Detail structure for the board
        //-----------------------------------------------------
---------------------
        unsafe public static Board_Details_T
GetBoardDetails(short hBoard)
        {
            Board_Details_T Board;

            Board.model = DIO_TC.GetBoardModel(hBoard);
            if (Board.model < 0)
            {
                Board.baseAddr = 0;
                Board.irq = -1;
                Board.isPci = false;
                Board.pciBus = -1;
                Board.pciSlot = -1;
                Board.hwVersion = 0;
```

```
                Board.cardName = "error";
                Board.cardDesc = "error";
            }
            else
            {
                Board.baseAddr = DIO_TC.GetBoardBase(hBoard);
                Board.irq = DIO_TC.GetBoardIRQ(hBoard);
                if (DIO_TC.GetBoardPciPosition(hBoard,
&Board.pciBus, &Board.pciSlot) == DIO_TC.OK)
                    Board.isPci = true;
                else
                    Board.isPci = false;
                DIO_TC.DIO_TC_hardwareVersion(hBoard,
&Board.hwVersion);
                switch (Board.model)
                {
                    case DIO_TC.PC212E:
                    case DIO_TC.PC214E:
                    case DIO_TC.PC215E:
                    case DIO_TC.PC218E:
                        {
                            if (Board.isPci)
                                Board.cardName = "PCI" +
(Board.model);
                            else
                                Board.cardName = "PC" +
(Board.model) + "E";
                        }
                        break;

                    case DIO_TC.PC36AT:
                        {
                            if (Board.isPci)  // It's really a
PCI236!
                                Board.cardName = "PCI236";
                            else
                                Board.cardName = "PC36AT";
                        }
                        break;

                    case DIO_TC.PC263:
                        {
                            if (Board.isPci)
                                Board.cardName = "PCI" +
(Board.model);
                            else
                                Board.cardName = "PC" +
(Board.model);
                        }
```

```
                        break;

                    case DIO_TC.PC24E:
                        {
                            // Could be a PC24E or PC25E
                            Board.cardName = "PC24E/PC25E";
                        }
                        break;

                    case DIO_TC.PC27E:
                        {
                            Board.cardName = "PC" +
(Board.model) + "E";
                        }
                        break;

                    case DIO_TC.PC26AT:
                    case DIO_TC.PC30AT:
                        {
                            Board.cardName = "PC" +
(Board.model) + "AT";
                        }
                        break;

                    case DIO_TC.PCI224:
                    case DIO_TC.PCI230:
                    case DIO_TC.PCI234:
                    case DIO_TC.PCI260:
                        {
                            Board.cardName = "PCI" +
(Board.model);
                            if (Board.hwVersion > 0)
                            {
                                Board.cardName += "+";
                            }
                        }
                        break;

                    default:
                        {
                            // Unknown card type
                            Board.cardName = "Type" +
(Board.model);
                        }
                        break;
                }  // end case

                Board.cardDesc = Board.cardName + " (base=" +
(Board.baseAddr).ToString("X") + "h irq=";
```

```
                if (Board.irq == DIO_TC.IRQ_NONE)
                    Board.cardDesc = Board.cardDesc + "None";
                else
                    Board.cardDesc = Board.cardDesc +
(Board.irq);

                if (Board.isPci)
                    Board.cardDesc = Board.cardDesc + "
pcibus=" + (Board.pciBus) + " pcislot=" + (Board.pciSlot);

                Board.cardDesc = Board.cardDesc + ")";
            }

            return Board;
        }

        unsafe public Board_Details_T GetBoardDetails()
        {
            return GetBoardDetails(hBoard);
        }

        //-----------------------------------------------------
---------------------
        // ReportError
        //-----------------------------------------------------
---------------------
        // Function
        //   Displays a Text box with the error description
        // Parameters
        //   The Error code
        // Returns
        //   None
        //-----------------------------------------------------
---------------------
        public void ReportError(short code)
        {
            string message;

            switch (code)
            {
                case DIO_TC.OK:
                    message = "Error code 0: OK!";
                    break;
                case DIO_TC.ERRSUPPORT:
                    message = "Error code -1: Board/feature
not supported";
                    break;
                case DIO_TC.ERRBASE:
```

```
                    message = "Error Code -2: Address already
registered";
                    break;
                case DIO_TC.ERRIRQ:
                    message = "Error Code -3: IRQ level
already registered";
                    break;
                case DIO_TC.ERRHANDLE:
                    message = "Error Code -4: Board/resource
not registered";
                    break;
                case DIO_TC.ERRCHAN:
                    message = "Error Code -5: Invalid
channel";
                    break;
                case DIO_TC.ERRDATA:
                    message = "Error Code -6: Invalid data";
                    break;
                case DIO_TC.ERRRANGE:
                    message = "Error Code -7: Out of range";
                    break;
                case DIO_TC.ERRMEMORY:
                    message = "Error Code -8: Insufficient
memory";
                    break;
                case DIO_TC.ERRBUFFER:
                    message = "Error Code -9: Buffer not
registered";
                    break;
                case DIO_TC.ERRPC226:
                    message = "Error Code = -10: PC226E not
found";
                    break;
                default:
                    message = "Error Code " + code.ToString()
+ ": Undefined code.";
                    break;
            } // end switch
            Console.WriteLine(message, "DIO_TC DLL Error");
        } // end sub


        //-----------------------------------------------------
---------------------
        // SetBoardList
        //-----------------------------------------------------
---------------------
        // Function
```

```
        //   Called by the user of the module to set up the
dropdown list of
        //   registerable boards before showing the form to
allow a board to be
        //   picked.
        // Parameters
        //   List of supported card types
        // Returns
        //   Mo, moified to return one card or collection
        //-----------------------------------------------------
---------------------
        public void SetBoardList(params short[] cardtypes)
        {
            int upperbt;
            short no = 0;
            bool boardExist = false;
            short oldResetOnRegister;

            // Call DIO_TC_SetResetOnRegister(0) to stop DLL
resetting hardware
            // on boards found by the following loop.
            // Driver versions prior to v4.40 will reset the
hardware regardless.
            //
            // N.B. Remove these two calls for DLL versions
prior to v4.40.
            oldResetOnRegister =
DIO_TC.DIO_TC_GetResetOnRegister();
            DIO_TC.DIO_TC_SetResetOnRegister(0);

            upperbt = cardtypes.Length;

            hBoard = DIO_TC.registerBoardEx(no);
            if (hBoard >= 0)
            {
                P_Boards[0] = GetBoardDetails(hBoard);
                DIO_TC.FreeBoard(hBoard);
                // Check board type is supported.
                // N.B. some PCI cards share the same model
number as a compatible
                // ISA card, e.g. PC215E could refer to PC215E
or PCI215.
                // This is the correct card slot:
Console.WriteLine("Card types: " + cardtypes(A. Petsa) + "
Card Model: " + P_Boards[0].model);
                if (P_Boards[0].model == cardtypes(A. Petsa))
                {
                    // Card type is supported by this program
                    //set to true, board found!
```

```
                    boardExist = true;
                }
            }

            // Restore previous 'reset on register' setting.
            // N.B. Remove this call for DLL versions prior to
v4.40.

DIO_TC.DIO_TC_SetResetOnRegister(oldResetOnRegister);

            if (boardExist)
            {
                //register the board
                hBoard =
DIO_TC.registerBoard(P_Boards[0].model, P_Boards[0].baseAddr,
P_Boards[0].irq);

                if (hBoard < 0)            // error detected
                    ReportError(hBoard);

                //otherwise print the card name we are
registering
                Console.WriteLine("Registered card name: " +
P_Boards[0].cardDesc);
            }
            else
            {
                // No compatible boards available for use.
                Console.WriteLine("NO SUPPORTED CARDS
AVAILABLE! Err300");
            }

        }

        public void registerMyBoard()
        {
            //init the BOARD REGISTER CLASS - BEGINS
            hBoard = DIO_TC.ERRSUPPORT;
            short[] cardtypes = new short[]
              {
                DIO_TC.PC26AT,
                DIO_TC.PC27E,
                DIO_TC.PC30AT,
                DIO_TC.PCI230,
                DIO_TC.PCI260
              };
            SetBoardList(cardtypes);
```

```
            G_SelectedCardHandle = hBoard; // Make a Local
Copy of the Cards Handle
            G_CardDetails = GetBoardDetails();

            if (G_SelectedCardHandle >= 0)
            {
                switch (G_CardDetails.model)
                {
                    case DIO_TC.PCI230:
                    case DIO_TC.PCI260:
                        {
                            if (G_CardDetails.hwVersion > 0)
                            {
                                // PCI230+ and PCI260+ have
16-bit ADC
                                UniScale = 65535.0;
                                BiScale = 32767.0;
                            }
                            //  panelPCI230.Visible = true;
                            // labelPCI230.Text =
G_CardDetails.cardName + " Settings";
                            // radioButtonPCI230_Bi_10.Checked
= true;
                            // radioButtonPCI230_SE.Checked =
true;
                            G_IntChip = DIO_TC.ADC2;
                        } break;
                }

                // For PC27E, put all three timer channel
outputs in the 'high' state by
                // setting them to mode 1. This is in case a
timer output is jumpered to
                // /TIM, as software triggers do not work on
the PC27E if /TIM is low.
                if (G_CardDetails.model == DIO_TC.PC27E)
                {
                    DIO_TC.TCsetMode(G_SelectedCardHandle,
DIO_TC.X2, 0, DIO_TC.MODE1);
                    DIO_TC.TCsetMode(G_SelectedCardHandle,
DIO_TC.X2, 1, DIO_TC.MODE1);
                    DIO_TC.TCsetMode(G_SelectedCardHandle,
DIO_TC.X2, 2, DIO_TC.MODE1);
                }

                // We are triggering conversions by software

DIO_TC.AIOsetADCconvSource(G_SelectedCardHandle, 0,
DIO_TC.CNV_SW);
```

```
                // Setup user interrupt
                StartInt();
            }
            else
            {
                // No Suitable Cards Found
                Console.WriteLine("No Suitable Cards Found -
Err301");
            }
            //some initialisation for digital I/O
            DIO_TC.DIOsetChanWidth(0, 0, 1);//set channel
width to 1 so have 24 I/O
            DIO_TC.DIOsetMode(0, 0, 0, 0);//set A0 to output


            //int the BOARD REGISTER CLASS - ENDS
        }

        private void StartInt()
        {
            cbProc = new DIO_TC.TTCCALLBACK(AIOProcessEvent);
            G_hTCInt =
DIO_TC.TCsetUserInterruptAIO(G_SelectedCardHandle, cbProc,
(IntPtr)0, G_IntChip, DIO_TC.ISR_READ_ADCS, 0, 1U <<
G_CurrentChannel);
            if (G_hTCInt >= 0)
            {
                // :-) everything seems to be OK so Enable the
Interrupts
                DIO_TC.enableInterrupts(G_SelectedCardHandle);
            }
        }

        private void AIOProcessEvent(short h, IntPtr wParam,
uint Dat)
        {
            double LocalData;

            // scale Data with whatever the Settings for the
Card have been Set to.
            LocalData = ((int)Dat * G_VoltageScale);
            LoadCellData = LocalData;

            //show current voltage and load at start
            MeterData = LocalData;
            currentLoadValue =
Calibration.getNewtons(MeterData);
```

```
            //Console.WriteLine("Voltage: " +
String.Format("{0:0.00}", MeterData));
            //Console.WriteLine("Current Load: " +
String.Format("{0:0.00}", currentLoadValue));
        }

        public void getAnalogueValue()
        {
            //this is a copy of the timer tick method
            short Polar;
            double PolarScale;
            double VScale;
            int Sel;
            uint GainBits;
            VScale = 4; // Stop The compiler whinning
            PolarScale = UniScale;
            GainBits = DIO_TC.PCI230_ADC_ALLRANGE20;

            // Set Settings depending on what card we have
            switch (G_CardDetails.model)
            {

                case DIO_TC.PCI230:
                case DIO_TC.PCI260:
                    {
                        Sel = G_CurrentPCI230_Range;
                        if (Sel < 4)
                        {
                            PolarScale = BiScale;
                            Polar = DIO_TC.ALLBIPOLAR;
                        }
                        else
                        {
                            PolarScale = UniScale;
                            Polar = DIO_TC.ALLUNIPOLAR;
                        }

                        switch (3)//use this to set scale for
load cell then can refactor
                        {
                            case 0: { VScale = 1.25; GainBits
= DIO_TC.PCI230_ADC_ALLRANGE2PT5; } break;
                            case 1: { VScale = 2.5; GainBits =
DIO_TC.PCI230_ADC_ALLRANGE5; } break;
                            case 2: { VScale = 5; GainBits =
DIO_TC.PCI230_ADC_ALLRANGE10; } break;
                            case 3: { VScale = 10; GainBits =
DIO_TC.PCI230_ADC_ALLRANGE20; } break;
```

```
                            case 4: { VScale = 2.5; GainBits =
DIO_TC.PCI230_ADC_ALLRANGE2PT5; } break;
                            case 5: { VScale = 5; GainBits =
DIO_TC.PCI230_ADC_ALLRANGE5; } break;
                            case 6: { VScale = 10; GainBits =
DIO_TC.PCI230_ADC_ALLRANGE10; } break;
                            case 7: { VScale = 20; GainBits =
DIO_TC.PCI230_ADC_ALLRANGE20; } break;
                        }
                        // Sets channel gains in hardware,
&HFFFF is channel mask - all 16 channels changed
                        // PCI Cards can set the Gain on the
cards withoiut Jumpers. The ISA cards
                        // have to phyically set the Gain with
Jumpers

DIO_TC.AIOsetHWADCchanGain(G_SelectedCardHandle, 0, 0xFFFF,
GainBits);
                    } break;
                default:
                    {
                        VScale = 1;
                        Polar = DIO_TC.ALLBIPOLAR;
                    } break;
            }

            // Sets polar mode in software and hardware (if
supported)
            DIO_TC.AIOsetAllADCchanMode(G_SelectedCardHandle,
0, Polar);

            // Set VoltageScale for AIOProcessEvent
            G_VoltageScale = (VScale / PolarScale);

            // Multiplexer was set up by the interrupt set-up
            // so start the AtoD
            DIO_TC.AIOstartADCconversion(G_SelectedCardHandle,
0);
        }
    }
}
```

6.2 Conference Abstract

Mohammad Ramezanian, Philip Trwoga, Ian Locke, and Andrew Afoke.

Title: Study of Behavior of Chondrocytes under chronic load that will result in the repair or destruction of cartilage. 4/12/2014, Faculty of Science & Technology.

Abstract: Interest in the mechanical properties of collagen fibers abundant in cartilage has been growing. Their complex mechanical properties often exhibit nonlinear, anisotropic, viscoelastic behavior over finite strains. Human cartilage is a non-innervated and avascular tissue. It is comprised of the highly-specialised and fibroblast-like chondrocytes and associated extracellular matrix (ECM). ECM Components are produced and maintained by the chondrocytes and include type 2 collagen, proteoglycans and non-collagenous proteins. In this study, multi-axis loading simulation hardware has been developed and will be used to study the behaviour of chondrocytes under chronic load. This will provide a better understanding of the ability of chondrocytes to modify the ECM.

References

A. PETSA, H. S. C., R.A. KNIGHT, A.N. AFOKE AND I.C. LOCKE 2004. Establishment and analysis of 3-D collagen gel cultures of C-20/A4 chondrocytes. *OARSI, University of Westminster*.

AIGNER T, H. M., NEUREITER D, GEBHARD PM, ZEILER G, KIRCHNER T, MCKENNA L. 2001. Apoptotic cell death is not a widespread phenomenon in normal aging and osteoarthritis human articular knee cartilage: a study of proliferation, programmed cell death (apoptosis), and viability of chondrocytes in normal and osteoarthritic human knee cartilage. *Arthritis Rheum.,* 44**,** 1304-12.

AMIN, A. R., ABRAMSON, S.B. 1998. The role of nitric oxide in articular cartilage breakdown in osteoarthritis. *Curr. Opin. Rheumatol.,* 10**,** 263-268.

ARCHER, C. M., H ; PITSILLIDES, AA 1994. CELLULAR ASPECTS OF THE DEVELOPMENT OF DIARTHRODIAL JOINTS AND ARTICULAR-CARTILAGE *Journal Of Anatomy,* 184**,** 447-456.

BI, X., YANG, X., BOSTROM, M.P.G., CAMACHO, N.P. 2006. Fourier transform infrared imaging spectroscopy investigations in the pathogenesis and repair of cartilage. *Biochimica et Biophysica Acta - Biomembranes,* 1758**,** 934-941.

BIRRELL, A. D. 2003. An introduction to programming with C# threads. Technical report. *Microsoft Corporation*.

BLANCO, F. J., GUITIAN, R., VÁZQUEZ-MARTUL, E., DE TORO, F. J. AND GALDO, F 1998. Osteoarthritis chondrocytes die by apoptosis: A possible pathway for osteoarthritis pathology. *Arthritis & Rheumatism***,** 284–289. doi: 10.1002/1529-0131(199802)41:2<284::AID-ART12>3.0.CO;2-T.

BUCKLEY, M. R., BERGOU, A. J., FOUCHARD, J., BONASSAR, L. J. & COHEN, I. 2010. High-resolution spatial mapping of shear properties in cartilage. *Journal of Biomechanics,* 43**,** 796-800.

BUCKWALTER JA, H. E., ROSENBERG L, ET AL. 1988. Articular Cartilage: Composition and Structure.

*Injury and Repair of the Musculoskeletal Soft Tissues.***,** 405-425.

BUCKWALTER, J. A., MANKIN, H.J 1998. Articular cartilage: tissue design and chondrocyte-matrix interactions. *Instructional Course Lectures,* 47**,** 477-486.

C.P. NEU, M. L. H., J.H. WALTON 2005. Heterogeneous three-dimensional strain fields during unconfined cyclic compression in bovine articular cartilage explants. *J. Orthop. Res.***,** 1390–1398.

CHEN, A. C., BAE, W. C., SCHINAGL, R. M., & SAH, R. L. 2001a. Depth-and strain-dependent mechanical and electromechanical properties of full-thickness bovine articular cartilage in confined compression. *Journal of biomechanics,* 34**,** 1-12.

CHEN, C.-T., BURTON-WURSTER, N., BORDEN, C., HUEFFER, K., BLOOM, S. E. & LUST, G. 2001. Chondrocyte necrosis and apoptosis in impact damaged articular cartilage. *Journal of Orthopaedic Research,* 19**,** 703-711.

CHEN, S. S., FALCOVITZ, Y. H., SCHNEIDERMAN, R., MAROUDAS, A., & SAH, R. L. 2001b. Depth-dependent compressive properties of normal aged human femoral head articular cartilage: relationship to fixed charge density. . *Osteoarthritis and Cartilage,* 9**,** 561-569.

CLEMENTS, K. M., BEE, Z. C., CROSSINGHAM, G. V., ADAMS, M. A. & SHARIF, M. 2001. How severe must repetitive loading be to kill chondrocytes in articular cartilage? *Osteoarthritis and Cartilage,* 9**,** 499-507.

DESCHNER J, H. C., PIESCO NP, AGARWAL S. 2003. Signal transduction by mechanical strain in chondrocytes. . *Curr Opin Clin Nutr Metab Care.,* 6**,** 289-293.

DI FEDERICO, E., BADER, D. L. & SHELTON, J. C. 2014. Design and validation of an in vitro loading system for the combined application of cyclic compression and shear to 3D chondrocytes-seeded agarose constructs. *Medical Engineering & Physics,* 36**,** 534-540.

F. GUILAK, R. L. S., L.A. SETTON 1997. Physical regulation of cartilage metabolism. *V.C. Mow, W.C. Hayes (Eds.), Basic Orthopaedic Biomechanics, Lippincott-Raven, Philadelphia* 179–207.

FRANK, E. H., JIN, M., LOENING, A. M., LEVENSTON, M. E. & GRODZINSKY, A. J. 2000. A versatile shear and compression apparatus for mechanical stimulation of tissue culture explants. *Journal of Biomechanics,* 33**,** 1523-1527.

FREEMAN, P., NATARAJAN, R., KIMURA, J. & ANDRIACCHI, T. 1994. Chondrocyte cells respond mechanically to compressive loads. *Journal of Orthopaedic Research,* 12**,** 311-320.

FREUTEL, M., SCHMIDT, H., DÜRSELEN, L., IGNATIUS, A. & GALBUSERA, F. 2014. Finite element modeling of soft tissues: Material models, tissue interaction and challenges. *Clinical Biomechanics,* 29**,** 363-372.

G.E. NUGENT-DERFUS, T. T., J.K. O'NEILL, S.B. CAHILL, S. GÖRTZ, T. PONG, H. INOUE, N.M. ANELOSKI, W.W. WANG, K.I. VEGA, T.J. KLEIN, N.D. HSIEH-BONASSERA, W.C. BAE, J.D. BURKE, W.D. BUGBEE, AND R.L. SAH, 2006. Continuous passive motion applied to whole joints stimulates chondrocyte biosynthesis of PRG4. *Osteoarthritis and Cartilage*.

GUILAK, F. & MOW, V. C. 2000. The mechanical environment of the chondrocyte: a biphasic finite element model of cell–matrix interactions in articular cartilage. *Journal of Biomechanics,* 33**,** 1663-1673.

GUPTA, S. V. 2012. Strain Gauge Load Cells. *In Mass Metrology, Springer Berlin Heidelberg.***,** 89-119.

HALL, A. C., J. P. G. URBAN, AND K. A. GEHL. 1991. The effects of hydrostatic pressure on matrix synthesis in articular cartilage. *Journal of orthopaedic research* 9**,** 1-10.

HALONEN, K. S., MONONEN, M. E., JURVELIN, J. S., TÖYRÄS, J. & KORHONEN, R. K. 2013. Importance of depth-wise distribution of collagen and proteoglycans in articular cartilage—A 3D finite element study of stresses and strains in human knee joint. *Journal of Biomechanics,* 46**,** 1184-1192.

HAN, S. K., FEDERICO, S., GRILLO, A., GIAQUINTA, G. & HERZOG, W. 2007. The Mechanical Behaviour of Chondrocytes Predicted with a Micro-structural Model of Articular Cartilage. *Biomechanics and Modeling in Mechanobiology,* 6**,** 139-150.

HELMINEN, H. J. J., J.; KIVIRANTA, I.; PAUKKONEN, K.; SAAMANEN, A-M.; TAMMI, M. 1987. Joint loading effects on articular cartilage: A historical review. In: Joint loading. *Bristol, England: Wright***,** 1–46.

HORTON JR, W. E., FENG, L. & ADAMS, C. 1998. Chondrocyte apoptosis in development, aging and disease. *Matrix Biology,* 17**,** 107-115.

HURTIG, M. B., NOVAK, K., MCPHERSON, R., MCFADDEN, S., MCGANN, L. E., MULDREW, K. E. N. & SCHACHAR, N. S. 1998. Osteochondral Dowel Transplantation for Repair of Focal Defects in the Knee: An Outcome Study Using an Ovine Model. *Veterinary Surgery,* 27**,** 5-16.

J. MIZRAHI, A. M., Y. LANIR, I. ZIV, T.J. WEBBER 1986. The “instantaneous” deformation of cartilage: effects of collagen fiber orientation and osmotic stress. *Biorheology,* 23**,** 311–330.

J.S. JURVELIN, M. D. B., E.B. HUNZIKER 2003. Mechanical anisotropy of the human knee articular cartilage in compression. *Proc. Inst. Mech. Eng. H***,** 215–219.

J.Z. WU, W. H. 2000. Finite element simulation of location- and time-dependent mechanical behavior of chondrocytes in unconfined compression tests. *Annals of Biomedical Engineering,* 28**,** 318–330.

JONES, W. R., PING TING-BEALL, H., LEE, G. M., KELLEY, S. S., HOCHMUTH, R. M. & GUILAK, F. 1999. Alterations in the Young’s modulus and volumetric properties of chondrocytes isolated from normal and osteoarthritic human cartilage. *Journal of Biomechanics,* 32**,** 119-127.

JURVELIN, J. S., BUSCHMANN, M. D., & HUNZIKER, E. B. 1997. Optical and mechanical determination of Poisson's ratio of adult bovine humeral articular cartilage. . *Journal of biomechanics,* 30**,** 235-241.

KÄÄB, M. J., ITO, K., CLARK, J. M. AND NÖTZLI, H. P. 1998. Deformation of articular cartilage collagen structure under static and cyclic loading. *J. Orthop. Res.,* 16**,** 743–751.

KNIGHT, M., GHORI, S., LEE, D. & BADER, D. 1998. Measurement of the deformation of isolated chondrocytes in agarose subjected to cyclic compression. *Medical engineering & physics,* 20**,** 684-688.
KONG, D., TIANSHENG ZHENG, MING ZHANG, DAODE WANG, SHIHAO DU, XIANG LI, JIAHU FANG, AND XIAOJIAN CAO. 2013. Static mechanical stress induces apoptosis in rat endplate chondrocytes through MAPK and mitochondria-dependent caspase activation signaling pathways. *PloS one* 7**,** e69403.
KORHONEN, R. K. & HERZOG, W. 2008. Depth-dependent analysis of the role of collagen fibrils, fixed charges and fluid in the pericellular matrix of articular cartilage on chondrocyte mechanics. *Journal of Biomechanics,* 41**,** 480-485.
KORHONEN, R. K., LAASANEN, M. S., TÖYRÄS, J., RIEPPO, J., HIRVONEN, J., HELMINEN, H. J. & JURVELIN, J. S. 2002. Comparison of the equilibrium response of articular cartilage in unconfined compression, confined compression and indentation. *Journal of Biomechanics,* 35**,** 903-909.
L.P. LI, J. S., M.D. BUSCHMANN, A. SHIRAZI-ADL 1999. Nonlinear analysis of cartilage in unconfined ramp compression using a fibril reinforced poroelastic model. *Clin. Biomech.,* 14**,** 673–682.
LEWIS, B., & BERG, D. J. 1995. hreads primer: a guide to multithreaded programming. *Prentice Hall Press*.
LOCKE, I. C., WHITE, O. B., TRWOGA, P. F. & AFOKE, A. N. 2010. 484 THE EFFECT OF INDENTOR SHAPE ON THE CREEP CURVES OF COLLAGEN GEL. *Osteoarthritis and Cartilage,* 18**,** S217-S218.
LUCCHINETTI, E., ADAMS, C. S., HORTON JR, W. E. & TORZILLI, P. A. 2002. Cartilage viability after repetitive loading: a preliminary report. *Osteoarthritis and Cartilage,* 10**,** 71-81.
M Y JAFFRIN, A. A. H. S. 1971. Peristaltic Pumping. *Annual Review of Fluid Mechanics,* 3**,** 13-37.
MAROUDAS, A. 1976. Balance between swelling pressure and collagen tension in normal and degenerate cartilage. *Nature,* 260**,** 808–809.
MICHAEL D. BUSCHMANN, Y. A. G., ALAN J. GRODZINSKY, ERNST B. HUNZIKER 1995. Mechanical compression modulates matrix biosynthesis in chondrocyte/agarose culture. *J Cell Sci,* 108**,** 1497-1508.
MOO, E. K., HAN, S. K., FEDERICO, S., SIBOLE, S. C., JINHA, A., ABU OSMAN, N. A., PINGGUAN-MURPHY, B. & HERZOG, W. 2014. Extracellular matrix integrity affects the mechanical behaviour of in-situ chondrocytes under compression. *Journal of Biomechanics,* 47**,** 1004-1013.
N. VERZIJL, J. D., C.B. ZAKEN, O. BRAUN-BENJAMIN, A. MAROUDAS, R.A. BANK, ET AL., 2002. Crosslinking by advanced glycation end products increases the stiffness of the collagen network in human articular cartilage – a possible mechanism through which age is a risk factor for osteoarthritis. *Arthritis Rheum,* 46**,** 114–123.
NAUMANN A, D. J., AWADALLAH A ET AL. 2002. Immunochemical and mechanical characterization of cartilage subtypes in rabbit. *J Histochem Cytochem,* 50**,** 1049–1058.
NGUYEN, Q. T., WONG, B. L., CHUN, J., YOON, Y. C., TALKE, F. E. & SAH, R. L. 2010. Macroscopic assessment of cartilage shear: Effects of counter-surface roughness, synovial fluid lubricant, and compression offset. *Journal of Biomechanics,* 43**,** 1787-1793.

O'CONNOR, S. M., STENGER, D. A., SHAFFER, K. M. & MA, W. 2001. Survival and neurite outgrowth of rat cortical neurons in three-dimensional agarose and collagen gel matrices. *Neuroscience Letters,* 304**,** 189-193.
PARK, S., HUNG, C. T. & ATESHIAN, G. A. 2004. Mechanical response of bovine articular cartilage under dynamic unconfined compression loading at physiological stress levels. *Osteoarthritis and Cartilage,* 12**,** 65-73.
PARK, S., KRISHNAN, R., NICOLL, S. B. & ATESHIAN, G. A. 2003. Cartilage interstitial fluid load support in unconfined compression. *Journal of Biomechanics,* 36**,** 1785-1796.
PFEILER, T. W., SUMANASINGHE, R. D. & LOBOA, E. G. 2008. Finite element modeling of 3D human mesenchymal stem cell-seeded collagen matrices exposed to tensile strain. *Journal of Biomechanics,* 41**,** 2289-2296.
PRINCE, D. E., AND JUSTIN K. GREISBERG. 2015. Nitric oxide-associated chondrocyte apoptosis in trauma patients after high-energy lower extremity intra-articular fractures. *Journal of Orthopaedics and Traumatology* 1-7.
R.A. BANK, M. S., A. MAROUDAS, J. MIZRAHI, J.M. TEKOPPELE 2000. The increased swelling and instantaneous deformation of osteoarthritic cartilage is highly correlated with collagen degradation. *Arthritis Rheum,* 43**,** 2202–2210.
R.M. SCHINAGL, D. G., A.C. CHEN, R.L. SAH 1997. Depth-dependent confined compression modulus of full-thickness bovine articular cartilage. *J. Orthop. Res.***,** 499–506.
RAMI, K. & SIMO, S. 2011. Biomechanics and Modeling of Skeletal Soft Tissues.
RAMI K. KORHONEN, M. S. L., JUHA TOYRAS, REIJO LAPPALAINEN, HEIKKI J. HELMINEN, JUKKA S. JURVELIN 2003. Fibril reinforced poroelastic model predicts specifically mechanical behavior of normal, proteoglycan depleted and collagen degraded articular cartilage. *J. Biomech.,* 36**,** 1373–1379.
SALTER, R. B. 1989. The Biologic Concept of Continuous Passive Motion of Synovial Joints: The First 18 Years of Basic Research and Its Clinical Application. *Clinical Orthopaedics and Related Research,* 242**,** 12-25.
SAUERLAND, K., RAISS, R. X. & STEINMEYER, J. 2003. Proteoglycan metabolism and viability of articular cartilage explants as modulated by the frequency of intermittent loading. *Osteoarthritis and Cartilage,* 11**,** 343-350.
SEEDHOM, B. B. & WALLBRIDGE, N. C. 1985. Walking activities and wear of prostheses. *Annals of the Rheumatic Diseases,* 44**,** 838-843.
SEIFZADEH, A., OGUAMANAM, D. C. D., TRUTIAK, N., HURTIG, M. & PAPINI, M. 2012. Determination of nonlinear fibre-reinforced biphasic poroviscoelastic constitutive parameters of articular cartilage using stress relaxation indentation testing and an optimizing finite element analysis. *Computer Methods and Programs in Biomedicine,* 107**,** 315-326.
SHIRAZI, R. & SHIRAZI-ADL, A. 2009. Computational biomechanics of articular cartilage of human knee joint: Effect of osteochondral defects. *Journal of Biomechanics,* 42**,** 2458-2465.
SONG J, D. F., LI X, ET AL 2014. Effect of Treadmill Exercise Timing on Repair of Full-Thickness Defects of Articular Cartilage by Bone-Derived Mesenchymal Stem Cells: An Experimental Investigation in Rats. *Huard J, ed. PLoS ONE. ,* 9**,** e90858. doi:10.1371.
SOPHIA FOX, A. J., BEDI, A., & RODEO, S. A. 2009. The Basic Science of Articular Cartilage: Structure, Composition, and Function. . *Sports Health,* 1**,** 461–468.

SUN, L., WANG, X. & KAPLAN, D. L. 2011. A 3D cartilage – Inflammatory cell culture system for the modeling of human osteoarthritis. *Biomaterials,* 32**,** 5581-5589.
TREPPO, S., KOEPP, H., QUAN, E. C., COLE, A. A., KUETTNER, K. E., & GRODZINSKY, A. J. 2000. Comparison of biomechanical and biochemical properties of cartilage from human knee and ankle pairs. *Journal of Orthopaedic Research,* 18**,** 739-748.
URBAN, J. P. G. 1994. THE CHONDROCYTE: A CELL UNDER PRESSURE. *British Journal of Rheumatology,* 33**,** 901-908.
V.C. MOW, W. Y. G., F.H. CHEN 2005. Structure and function of articular cartilage and meniscus. *Basic Orthopaedic Biomechanics and Mechano-biology., Lippincott Williams & Wilkins, Philadelphia***,** 720.
W. HERZOG, S. D., E. SUTER, P. MAYZUS, T.R. LEONARD, C. MULLER, ET AL. 1998. Material and functional properties of articular cartilage and patellofemoral contact mechanics in an experimental model of osteoarthritis. *J. Biomech.,* 31**,** 1137–1145.
W. ZHU, V. C. M., T.J. KOOB, D.R. EYRE 1993. Viscoelastic shear properties of articular cartilage and the effects of glycosidase treatment. *J. Orthop. Res.***,** 771–781.
WATANABE, H. 2015. Chondroitin Sulfate in Cartilage. *In:* TANIGUCHI, N., ENDO, T., HART, G. W., SEEBERGER, P. H. & WONG, C.-H. (eds.) *Glycoscience: Biology and Medicine.* Springer Japan.
WERNIKE, E., LI, Z., ALINI, M. & GRAD, S. 2008. Effect of reduced oxygen tension and long-term mechanical stimulation on chondrocyte-polymer constructs. *Cell and Tissue Research,* 331**,** 473-483.
WONG, B. L., KIM, S. H. C., ANTONACCI, J. M., MCILWRAITH, C. W. & SAH, R. L. 2010. Cartilage shear dynamics during tibio-femoral articulation: effect of acute joint injury and tribosupplementation on synovial fluid lubrication. *Osteoarthritis and Cartilage,* 18**,** 464-471.
WONG, B. L. & SAH, R. L. 2010. Mechanical asymmetry during articulation of tibial and femoral cartilages: Local and overall compressive and shear deformation and properties. *Journal of Biomechanics,* 43**,** 1689-1695.
WU, J. Z., HERZOG, W. & EPSTEIN, M. 1999. Modelling of location- and time-dependent deformation of chondrocytes during cartilage loading. *Journal of Biomechanics,* 32**,** 563-572.
YUSOFF, N., ABU OSMAN, N. A. & PINGGUAN-MURPHY, B. 2011. Design and validation of a bi-axial loading bioreactor for mechanical stimulation of engineered cartilage. *Medical Engineering & Physics,* 33**,** 782-788.
ZHANG, M., ZHENG, Y. P. & MAK, A. F. T. 1997. Estimating the effective Young's modulus of soft tissues from indentation tests—nonlinear finite element analysis of effects of friction and large deformation. *Medical Engineering & Physics,* 19**,** 512-517.
F. N. Ghadially, Fine structure of synovial joints, London, Butterworthand COLtd, 1983.

L. Eichelberger, J. S. Miles and M.Roma, “The histochemical characterization of articular cartilage of poliomyelitis patients,” Journal of Laboratory and Clinical Medicine; vol 40, pp. 284-96, 1952.

K. S. Kostenszky, E. H. Olah, “Effect of increased hctional demand on the glucosaminoglycan (mucopolysaccharide) content of the articular cartilage,” Acta Biologica Academiae Scientiarum Hungaricae. V0123(1), pp. 75-82, 1972.

J. P. Paul, “Force actions transmitted by joints in the human body,” Proceedings of the Royal Society of London, Series B: Biological Sciences. Vol. 192(1107), pp. 163-72, 1976.

C. G. Armstrong, A. S. Bahrani and D. L. Gardner, “In vitro measurement of articular cartilage deformations in the intact human hip joint under load,” Journal of Bone & Joint Surgery, AmericanVol.61(5),pp.744-55,1979

Guilak, F., et al., The role of biomechanics and inflammation in cartilage injury and repair. Clinical Orthopaedics and Related Research, 2004(423): p. 17-26.

Visse, R. and H. Nagase, Matrix metalloproteinases and tissue inhibitors of metalloproteinases - Structure, function, and biochemistry. Circulation Research, 2003. 92(8): p. 827-839.

Goldring, M.B. and S.R. Goldring, Osteoarthritis. Journal of Cellular Physiology, 2007. 213(3): p. 626-634.

Palmer, A.W., et al., Composition-function relationships during IL-1- induced cartilage degradation and recovery. Osteoarthritis and Cartilage, 2009. 17(8): p. 1029-1039.

DeLise, A.M., L. Fischer, and R.S. Tuan, Cellular interactions and signaling in cartilage development. Osteoarthritis and Cartilage, 2000. 8(5): p. 309-334.

Hartmann, C. and C.J. Tabin, Dual roles of Wnt signaling during chondrogenesis in the chicken limb. Development, 2000. 127(14): p. 3141-3159.

Goldring, M.B., K. Tsuchimochi, and K. Ijiri, The control of chondrogenesis. Journal of Cellular Biochemistry, 2006. 97(1): p. 33- 44.

Roman-Blas, J.A. and S.A. Jimenez, NF-kappa B as a potential therapeutic target in osteoarthritis and rheumatoid arthritis. Osteoarthritis and Cartilage, 2006. 14(9): p. 839-848.

Liacini, A., et al., Inhibition of interleukin-1-stimulated MAP kinases, activating protein-1 (AP-1) and nuclear factor kappa B (NF-kappa B) transcription factors down-regulates matrix metalloproteinase gene expression in articular chondrocytes. Matrix Biology, 2002. 21(3): p. 251-262.

Kitano, H., Systems biology: A brief overview. Science, 2002. 295(5560): p. 1662-4.

Boccaletti, S., et al., Complex networks: Structure and dynamics. Physics Reports-Review Section of Physics Letters, 2006. 424(4-5): p. 175-308.

Aggarwal, K., K.H. Lee, Functional genomics and proteomics asa foundation for systems biology. Brief Functional Genomic Proteomic, 2003. 2(3): p. 175-84.

Ioannis N. Melas, Aikaterini D. Chairakaki, Alexander Mitsos, Zoe Dailiana, Christopher G. Provatidis, Leonidas G. Alexopoulos, Modeling signaling pathways in articular cartilage, 33rd Annual International Conference of the IEEE EMBS Boston, Massachusetts USA, August 30 - September 3, 2011