Design and Characterization of a Gel Loading Mechanism for an Ultra-High Throughput Mutational Spectrometer

by

Nathan B. Ball

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2005

Signature of Author_____
Department of Mechanical Engineering
May 10, 2004

Certified by_____
Ian W. Hunter
Hatsopoulos Professor of Mechanical Engineering
Thesis Supervisor

Accepted by_____
Ernest G. Cravalho
Chairman of the Undergraduate Thesis Committee, Mechanical Engineering

Design and Characterization of a Gel Loading Mechanism for an Ultra-high Throughput Mutational Spectrometer

by

Nathan B. Ball

Submitted to the Department of Mechanical Engineering
on May 10, 2005 in partial fulfillment of the requirements
for the Degree of Bachelor of Science in Mechanical Engineering

ABSTRACT

A process known as Constant Denaturant Capillary Electrophoresis is used to separate mutant from wild-type DNA at fractions down to 10^{-7}. A device known as an Ultra-high Throughput Mutational Spectrometer is being created to run 10,000 parallel channels of CDCE in order to correlate multiple point mutations in DNA with the diseases that they can cause, such as cancer. By separating the DNA in large populations, the underlying causes of such diseases can be identified. To successfully run CDCE, a high viscosity polymer gel must be loaded into each of the 10,000 channels, each of which are composed of an individual glass capillary with a 75 μm inner diameter. A mechanism was designed and tested which loaded gel into 8 channels simultaneously. The mechanism was used to test the relationship between gel loading time in relation to varied pressure and capillary length, through 45 total runs, with 8 channels per run. The relationships were characterized, resulting in two equations that enable an accurate prediction of the fill time necessary to load 10,000 parallel channels simultaneously under varied conditions.

Thesis Supervisor: Ian W. Hunter

Title: Hatsopoulos Professor of Mechanical Engineering

Acknowledgements

First and foremost, I would like to thank Prof. Ian Hunter for including me in his BioInstrumentation lab. My experience in his lab has been one of unparalleled inspiration and intellectual stimulation, and it is an honor to work with such an incredible group of researchers.

Also, thanks to Craig Forest, Tim Fofonoff, Bryan Schmid, Grant Kristofek, and everyone else at the BioInstrumentation Lab for their guidance, support and friendship.

Lastly, thanks to my incredible family for the support and love through all the projects on the way to MIT: go-karts, potato guns, 1,000,000 v Tesla coils, and even burning down the kitchen with rocket fuel. I could not have done it without you.

Table of Contents

Table of Figures

1.0 Introduction

Detection of mutated genes within the human genome is becoming an increasingly important endeavor. Rising numbers of human disease genes are being identified, along with growing numbers of disease-causing mutations within the genome [1]. Many methods are in use today that successfully detect genetic mutations, but with varying limits of throughput and accuracy. The aim of this project is to create an Ultra-high Throughput Mutational Spectrometer (UTMS) which is capable of separating and isolating genetic mutations at a rate several orders of magnitude higher than the best contemporary machines. By running 10,000 parallel channels of mutation detection at once, the UTMS will enable entire population bases to be analyzed in a matter of weeks, quickly and accurately correlating combinations of genetic mutations with specific diseases [6]. The process by which the separation of DNA is performed requires a high-viscosity polymer gel to be loaded into each of the 10,000 channels. The goal of this work was to create and characterize a mechanism that would accomplish that task.

2.0 Genetic Mutations

The double helix of the DNA strand which comprises the human genome is composed of four base molecules; adenine, thymine, cytosine and guanine. These four molecules, referred to by their initials A, T, C and G, bond together to form base pairs: A to T, and G to C [9]. A point mutation in the genome is a mismatch in a single base pair of DNA; for instance, an A that has bonded with a C instead of a T. When a gene that encodes a particular protein includes a mismatched base pair, the protein may not be produced properly by the cell, potentially causing disease within the individual.

The problem of genetic mutation is significant because mutations can be inherited. During cell reproduction, the linkage of chromosomes tends to cause genes in close regions to be inherited together [9]. These regions are known as hot spots. If a genetic mutation is located in a hot spot, it will likely be passed on to the new cell during meiosis. If an individual inherits a particular set of genomic mutations that are responsible for a disease, he will be genetically predisposed to contracting that disease in the future [9]. Though many other types of mutations and mutation-caused diseases exist, the UTMS project aims to identify specifically the diseases caused by single and multiple point mutations, both within the same gene, and across different genes.

2.1 Mutation Detection

Point mutations can occur in human DNA in fractions lower than 1/10,000,000 (10^{-7}) [1]. Of the 3×10^{9} base pairs in the entire genome, this means that fewer than 0.33% of the base pairs are likely to be mismatched. High-fidelity mutation detection is required to identify and separate mutated DNA segments from wild-type (un-mutated) segments for analysis. One technique suitable for this is called Denaturing Gradient Gel Electrophoresis (DGGE) [7].

DGGE utilizes the difference in electrophoretic mobility between mutant and wild-type segments for separation. Electrophoresis is performed by placing DNA samples in a high-voltage electric field. Since DNA carries a net negative charge, the strands are pulled toward the positive end of the field. The electrophoresis is carried out in a medium

of polymer gel, whose polymer matrix creates a molecular sieve. The DNA samples travel through this sieve toward the cathode.

Separation between the mutant and wild-type strands occurs due to the reduced electrophoretic mobility of the mutant strands compared to the wild-type strands: when DNA is heated to a high temperature (typically near 80 °C, depending on the sequence), the molecular bonds begin to break between base pairs in the DNA, and the double helix splits apart, or denatures. Because the molecular bond of a mismatched pair is weaker than that of a wild-type base pair, less energy is required to break the bond. Thus, a mutated DNA strand will denature at a lower temperature than its wild-type counterpart [7]. As a DNA strand denatures, its "arms" spread apart and get caught in the polymer gel matrix, inhibiting its motion toward the high-voltage cathode. If a mixture of mutant and wild-type is brought to a temperature between the denaturing temperatures of both strands, the mutant strands will denature and slow, while the wild-type strands maintain their speed toward the cathode [9].

By attaching a phosphorescent molecule to every strand, a gradient can be observed via fluorescence detection which clearly shows a spatial gradient formed by the uneven mobility of the strands [7]. This gradient in a DGGE gel is shown in Figure 1, below.

Figure 1: Separation of mutant and wild-type DNA in DGGE [10]

An improvement on DGGE is a process known as Constant Denaturing Capillary Electrophoresis (CDCE) [7]. Instead of performing the separation in a large gel slab, a quartz capillary with a small inner diameter (75 μm) is filled with the polymer gel. Conducting the separation across a quartz capillary rather than a gel slab allows for a higher voltage, increasing the speed of the entire process. The small thermal mass of the capillary compared to a gel slab allows for precise temperature control as well. Thus, instead of approximating the separation temperature and observing a spatial DNA separation gradient, the system can be run at a precise temperature, and the separation is instead observed through time [7]. By measuring the phosphorescent signal through a window in the capillary, five clear signal peaks will occur—two homoduplexes (both sides of the double helix are either mutated wild type), two heteroduplexes (a double helix with one wild type and one mutated strand), and one peak for the primers. The primers arrive first. The wild-type homoduplex is next, followed by the mutated homoduplex, and the heteroduplexes are last. A separation signal from a CDCE run is shown in Figure 2, below.

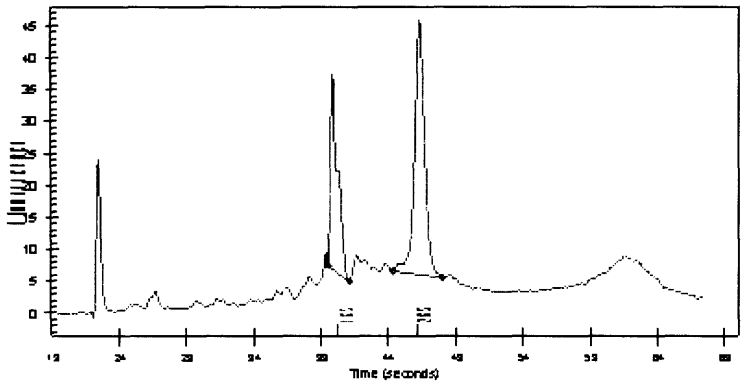

Figure 2: Three of the five separation peaks of mutant and wild-type DNA in CDCE [11]

The time gradient can be used to separate wild-type and mutant samples in separate channels as well. Two capillaries can be run simultaneously, one with a mutant sample and one with a wild-type sample. The wild-type signal will appear first, revealing which sample is mutated and which is not. This capability is highly useful because it allows for mutagenic analysis without prior phenotypic selection [5]—the basis for mutational spectrometry, and the greater goal of this project.

2.2 Mutational Spectrometry

The capability enabled by CDCE to separate mutant from wild-type DNA segments based on temperature facilitates the creation of a mutational spectrum: a set of data that correlates specific point mutations and combinations of mutations with the diseases that they cause [6]. With CDCE, phenotypic selection is not necessary to create a mutational spectrum [5]. This means that mutated DNA can be identified in a set of people independently of traits that might indicate its presence, such as test subjects with cancer. DNA segments from millions of individuals can be compared against one another, and

the phenotypes can be correlated with the results of the mutational spectral scan to gather information about specific mutations and their associated diseases [6].

To create a large mutational spectrum that spans as much of the genome as possible and identifies the resulting diseases of as many mutations and mutational combinations as possible, a massive amount of data is required. Current methods of sequencing and scanning only allow throughputs of up to several hundred channels at a time. With these methods, a prohibitively large amount of time and costs is required to gather the data necessary to be useful in population genetics. The goal of this large-scale project is to create a massively parallel DNA scanning machine capable of up to 10,000 parallel channels of mutational analysis. With such a device, the data throughput will enable genetic population analysis to reveal the underlying causes of many forms of cancer and other diseases.

3.0 Instrument Concept

The Ultra-high Throughput Mutational Spectrometer (UTMS) is an instrument that will conduct simultaneous mutational scanning of 10,000 different individuals via 10,000 parallel channels of CDCE. The mutational sensitivity of each channel will detect mutations at fractions of 10^{-7}, allowing for the detection of extremely rare mutations and combinations of mutations. To accomplish successful separation in such a massive instrument, several significant design innovations have been achieved.

To maintain precise (±0.1 °C) temperature resolution, a large liquid flow chamber encloses all of the 100 × 100 capillaries. This chamber contains cross-flow impedance blocks, which maintain orthogonal flow to the capillaries along most of their 300 mm length. For a high insulative value to allow high voltage separation, and to minimize thermal mass and maximize heat dissipation due to joule heating, the capillaries are small-diameter quartz tubes coated with polyimide. The inner diameter is 75 μm, and the outer 350μm. A picture of the tip of one capillary is shown below, in Figure 3.

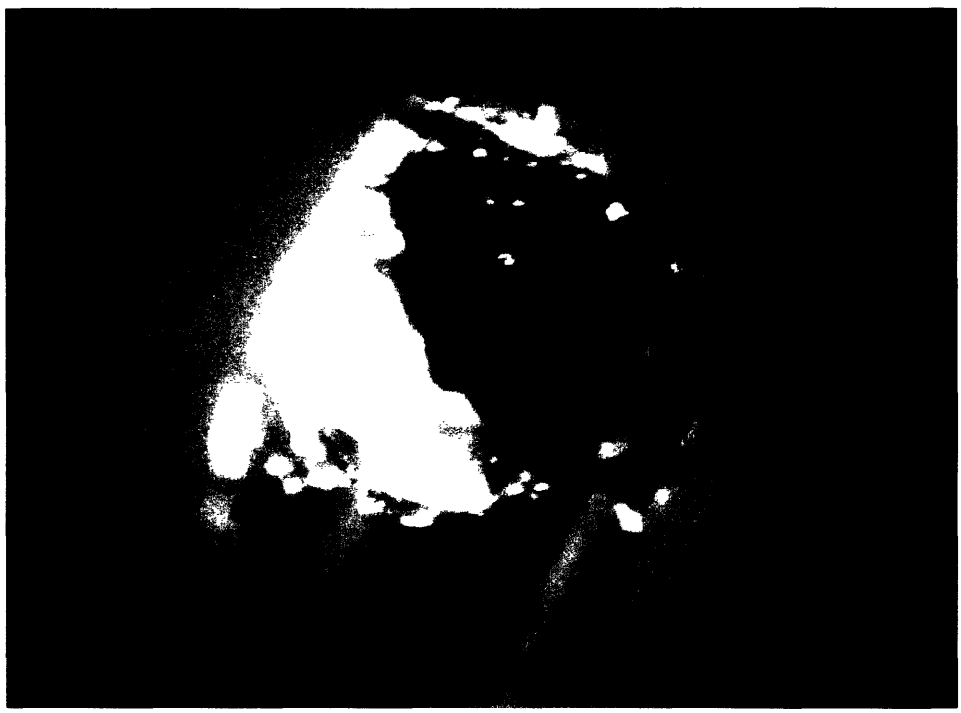

Figure 3: Cleaved quartz capillary with polyamide coating, 75 μm ID, 350 μm OD, made by Polymicro.

Fluorescence detection is typically accomplished for CDCE by shining a 488 nm laser normal to the capillary through a window in the polyimide coating. With a 100 × 100 array of capillaries, this is impossible. Two-dimensional end-on fluorescence

detection is an alternative design to collect signals from each capillary in such an array. To achieve the intensity of fluorescence excitation necessary for 10,000 capillaries, lasers are impractical and expensive. Fluorescence excitation is accomplished with a 170 W array of forty-two ultra-bright LEDs which shine directly down onto the tips of the capillaries. Instead of a one-dimensional photomultiplier tube, a high-sensitivity impact ionization CCD is used to observe the array end-on, and collect the data via fluorescence imaging through time.

To conduct electrophoresis across the capillaries, the tip of each channel must be in contact with the high voltage cathode. And to be able to detect fluorescence at the tip of the capillaries, the tip must be visible to the CCD which collects the fluorescent signals. This means that the electrode must not block the view, and yet must still contain conductive fluid buffer to complete the circuit.

A micro-manufactured array of wells with a silicone sealer has been created to accomplish this task. A clear window constitutes the top layer, with a ring-shaped electrode that encircles the capillary without obstructing the view of the tip. A layer of silicone seals around the capillary at the bottom layer, preventing evaporation of the fluid buffer. To increase the intensity of 488 nm excitation from the LED array, a micro-manufactured LED array is placed over the buffer wells. With one lens over each buffer well, the signal to noise ratio is increased by over 250×. The lens both multiplies the excitation intensity and enhances signal viewing to the CCD by magnifying the

fluorescent image. A diagram of a single buffer well and lens setup is shown below in Figure 4.

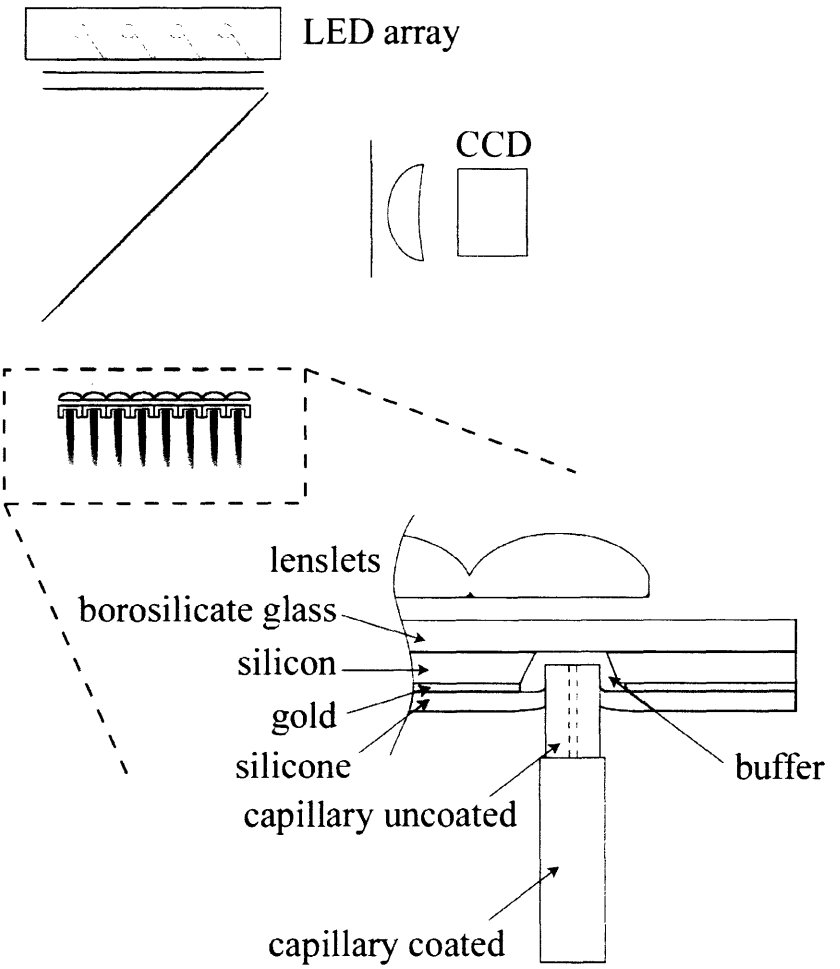

Figure 4: Micro-manufactured buffer well and micro-lenslet above capillary [12].

A critical component of the entire system is the mechanism to load the polymer gel separation matrix into the 75 μm inner diameter of each of the 10,000 capillaries. This is the purpose of this thesis: to design, analyze and construct a device to load gel into the

UTMS for all 10,000 capillaries, and to use it to characterize the loading performance as affected by temperature, capillary length, and system pressure.

The gel loader is placed below the entire system, and can accommodate the DNA sample loading card which is used during actual analysis. This is shown at the bottom of the diagram in Figure 5, below. Above the system is the optical data acquisition setup, consisting of the LED array, light collimators, excitation filter, dichroic beam splitter, and high-sensitivity impact ionization CCD. The entire system appears as follows in Figure 5.

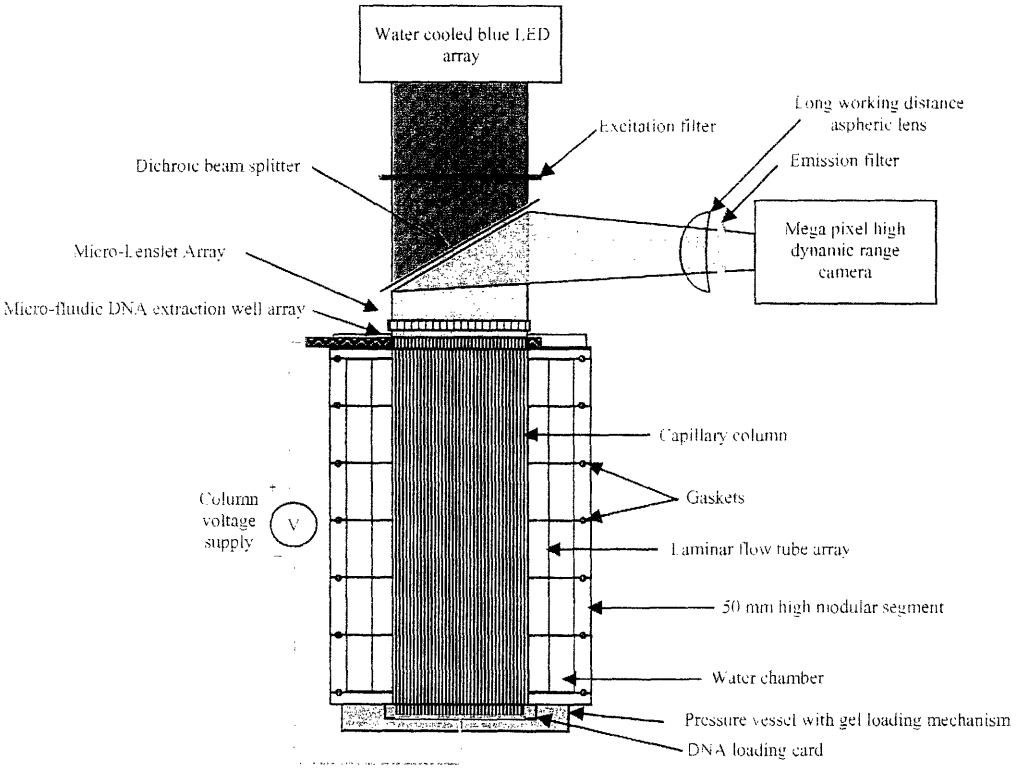

Figure 5: Ultra-high Throughput Mutational Spectrometer [13].

3.1 Design Requirements for Gel Loader

The gel loading mechanism for the UTMS is a simple device which accomplishes the task of loading polymer gel into the full 300 mm length of each of the 10,000 capillaries. The mechanism must load gel reliably and in a controlled manner, with minimal waste. Additionally, the mechanism must be able to accommodate the 10,000 well DNA loading card from which the DNA segments are inserted into the channels, and be able to switch out the gel for capillary wash in order to clean out and reinsert new gel into the capillaries.

Additionally the mechanism must be able to be temperature controlled, in the case that a heat-thinning polymer gel is used and its dynamic viscosity must be reduced to adequately load all of the channels. The gel loading system must be modular in order to accommodate different polymer matrix gels and capillary washes, and must be removable completely in order to allow a plugged capillary to be replaced from the device. Additionally, the gel loader must interface with the capillary clamp at the lower end which maintains a hold on the capillaries during use. This interface must be sealed to pressures greater than 200 kPa in order to safely load gels of very high viscosity in an acceptable period (<30 min) of time.

Two final goals of this apparatus are that it be a scalable prototype capable of loading 100 simultaneous capillaries, and proves the concept and technology for loading all 10,000 simultaneously, and that it be usable outside the UTMS. Usability outside the UTMS is important for two reasons: first, that the mechanism can quickly and easily load

small numbers of capillaries (down to a single capillary) for individual channel testing, and second, that it be able to do many runs in a row for characterization purposes.

A parallel objective for this device and this thesis is to use the scaled-down gel loader to perform a statistical analysis of different loading conditions, and to characterize the statistically distributed arrival time of gel into each capillary and how it varies in relation to loading pressure, capillary length, and system temperature. With a characterization of the gel loading times in terms of these factors, an expectation can be set for the percentage of the 10,000 capillaries which are full after a given amount of time.

To do this, a mathematical model was first constructed to predict the loading time of a capillary under the given conditions. Then the mathematical model was experimentally verified using the apparatus under various changes of those conditions, and a statistical distribution of loading times was measured for each set of conditions. With this information, a directly scaled statistical expectation of gel loading performance for all of the 10,000 capillaries was constructed, which will be useful in testing and use of the fully scaled, 10,000 channel UTMS.

4.0 Theory

4.1 Viscous Fluid Model of Gel Loader

In order to describe the velocity and subsequently the loading time of polymer gel into the capillary under given conditions of temperature and pressure, an equation was needed which correlated the viscous behavior of the gel in relation to those given conditions. The energy equation for steady incompressible flow [2] was used, as follows:

$$p_1 + \rho\frac{V_1^2}{2g} + \rho z_1 = p_2 + \rho\frac{V_2^2}{2g} + \rho z_2 + h_L, \qquad 1$$

where p is the pressure, ρ is the fluid density, V is the flow velocity, g is the acceleration of gravity, z is the height, and h_L is the head loss, or energy loss in the flow due to viscous shear. The Darcy-Weisbach Equation [2], determines the major head loss in the conduit as

$$h_L = f\frac{l}{D}\frac{V^2}{2g}, \qquad 2$$

where h_L is the head loss, l is the pipe length, D is the pipe diameter, V is the fluid velocity, g is the acceleration of gravity, and f is known as the Darcy Friction Factor [2], which for laminar flow in a smooth pipe is

$$f = \frac{64}{\mathrm{Re}}, \qquad 3$$

where Re is the Reynolds number, as

$$\mathrm{Re} = \frac{\rho VD}{\mu}, \qquad 4$$

where ρ is the fluid density and μ is the fluid viscosity. Because of the flow geometry from the gel bath into the tip of the capillary, there is a minor head loss term of $K_L = 0.8$. Minor head loss h_L is described as

$$h_L = K_L \frac{V^2}{2g}, \tag{5}$$

where K_L is a coefficient corresponding to the inflow geometry of the capillary in the gel bath. Thus, for the total head loss of the system, Figures 1 and 4 are added to yield

$$h_{L\,total} = K_L \frac{V^2}{2g} + f \frac{l}{D} \frac{V^2}{2}. \tag{6}$$

With all terms defined, Figure 1 was rearranged in order to solve for the velocity in terms of the other parameters, viz.

$$V = \sqrt{\frac{\Delta p - \rho \Delta z}{\frac{K_L}{2g} - f\frac{l}{D}}}. \tag{7}$$

Since the velocity was being used to ultimately calculate the time required to fill the capillary, it was necessary to find the change in velocity as the gel progressively filled its tube. To do this, the velocity was calculated in MATLAB at 1mm intervals all along the length of the capillary. Then each velocity was multiplied by its length segment, 1mm, to get the amount of time spent in each section. The times were added up, and the total fill time was thus calculated.

For this model to be accurate, it was necessary to verify that the turbulent entry length at the beginning of the capillary was not long enough to nullify the fully developed flow condition assumed for the validity of the above derivations. To calculate the entry length,

the approximate entry length equation for laminar flow was utilized. First, it was verified that the Reynolds number was below 2100 to make sure the flow was laminar. Then the equation

$$\frac{L_e}{D} \cong 0.06\,\mathrm{Re}_D \qquad \mathbf{8}$$

was used, where L_e is the entry length. By this calculation, the entry length was found to be less than 0.1 mm for the shortest length of capillary. Thus, the entry length was negligible, and the flow could be considered fully developed along its entirety.

In order to find measure the relationship between injection time and varied length and pressure, three measurements were taken for each parameter: one baseline measurement, and two additional for both length and pressure. The data points for the average fill times of each parameter were graphed, and a least squares line was fit to the data. With the equation of that line, predictions could be made for the performance of the gel loader when varying the conditions even further.

4.2 t-Test Statistical Analysis of Results

Due to the non-feasibility of testing on the same scale of runs as an ultra-high throughput gel load, the statistical performance of the gel loader must be analyzed and extrapolated approximately two orders of magnitude forward in order to predict the performance of an actual 10,000 channel parallel load. In order to calculate the accuracy of a smaller statistical sample size, some analysis of the data is necessary. To do this, the significance of each parameter is measured between sets of data using the t-test. This is a method of calculating the significance of the difference in means between two varied

groups. Though it is especially useful to compare the means of two randomly sampled groups to determine significance of variability, it is directly applicable to measuring the differences between gel loads in varying pressure and length.

An analogy of the t-test is a signal to noise ratio. The difference in means between the two groups is effectively the signal, as illustrated below in Figure 6. The "noise" is the variability within those groups, which may make it harder to see that difference.

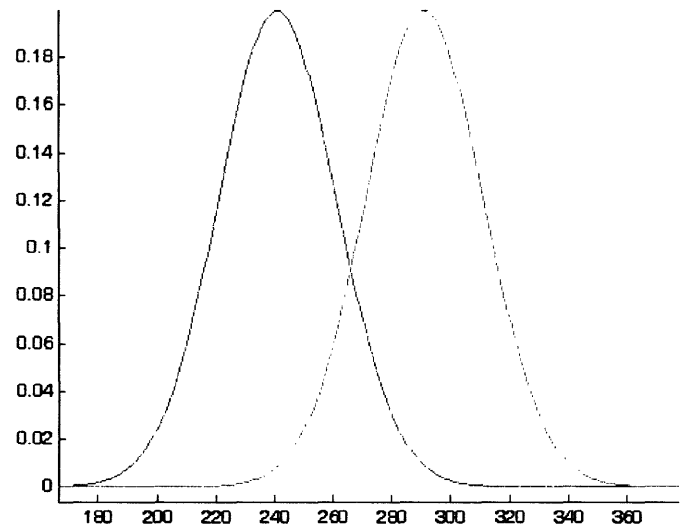

Figure 6: Illustration of two signal probability density functions and the difference between their means.

The equation to determine "t" in the t-test is shown below as Equation 10:

$$\frac{signal}{noise} = \frac{difference\ \ between\ \ group\ \ means}{\text{var}iability\ \ of\ \ groups} = t\,, \qquad \textbf{11}$$

which is mathematically calculated as

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{\text{var}_1}{n_1} - \frac{\text{var}_2}{n_2}}}, \qquad \textbf{12}$$

where $\overline{X}_1$ and $\overline{X}_2$ are the means of groups 1 and 2, var_1 and var_2 are the variability of groups 1 and 2, i.e. the squares of the standard deviations, and n_1 and n_2 are the sizes of groups 1 and 2. Once the ratio t is found, a standard t-table from any statistics book can be consulted to determine the percentage dictating the range within which the actual mean samples can be found [14].

By using the t-test between each of the sample sets taken during gel runs, the significance of the variance can be found between the groups being compared. This is most useful to compare runs within a data set against one another, to determine how accurate the run is with respect to the actual mean of the data, as calculated by the t-test.

However, in order to execute a t-test between every run within every dataset, a massive number of t-tests are required, with significant data sorting afterward to make meaningful use of the testing. ANOVA (ANalysis Of VAriance) is a method which gathers all the data that would be acquired by multiple t-tests and puts them into a single number, F (the found variation of the group averages divided by the expected variation), and gives a probability for the null hypothesis for the entire data set as p [8]. By using one-way ANOVA within the varying pressure and varying length data runs, the p value can be calculated that indicates the significance of the independent variable. A p value below 0.05 indicates that the particular variable is significant [15].

5.0 Prior Art in Polymer Gel Loading

In single capillary CDCE, a syringe is used to inject gel into the small-bore quartz capillary. Because high viscosity gels are sometimes used, the pressure necessary to inject the gel through the length of the capillary can get very high if the process is to be done quickly. In order to make a seal between the syringe and the capillary that is capable of handling the necessary pressures, a small Teflon tube is stretched over the ends of both the capillary and the syringe, as shown in Figure 7 below.

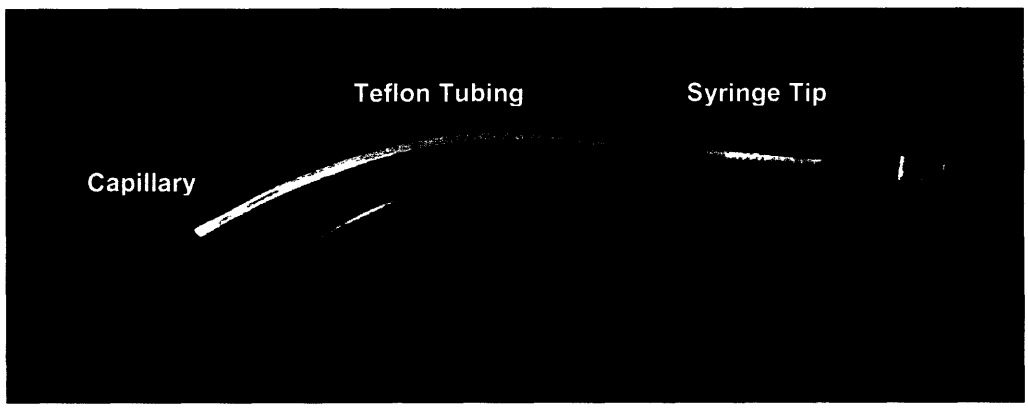

Figure 7: Teflon tubing is stretched over a syringe tip and a capillary to seal for gel injection.

This is clearly impractical for 10,000 capillaries, as it is difficult to create such a seal for a single capillary, much less thousands. Current state of the art for multi-channel DNA sequencers and scanners is to dip the ends of the capillaries (the most of which are 384 in the largest scanner to date) into a bath of gel within a pressure vessel, and utilize high-pressure inert gas such as nitrogen to impose a pressure differential across the capillary.

A notable and disadvantageous feature of these systems is that the capillaries are potted permanently within a hard resin to create an environmental seal for the pressure vessel. If a capillary becomes plugged in such a system, it cannot be replaced. With the high likelihood of many non-functional capillaries in a 10,000 channel system, it is imperative to be able to replace the capillaries. Because of this, it is necessary for the gel injection pressure vessel to accommodate a modular and replaceable capillary clamping mechanism that enables capillary replacement when necessary. This was taken into consideration for the design of the UTMS gel loader.

One other experimental version utilizes a radial configuration of capillaries which have been etched into a silicon wafer by micro-manufacturing. The inner radius of the array is an open tube, to which all the etched channels are exposed. Gel or capillary wash are fed into the open tube via a valve system, and are forced through the etched channels at over 6.9 MPa (1000 PSI) [4]. A major drawback of this etched plate design is the limited number of channels it can accommodate. The version presented for micro-manufacturing could hold a maximum of 96 channels. The device designed in this thesis is scalable to 10,000 channels, and beyond. Additionally, the problem of replacing a capillary is still faced with this system. With etched channels, if one clogs, it is ruined. With the interface to the replaceable capillary system, the gel loader is a powerful and versatile tool.

5.1 Design of UTMS Scalable Gel Loader, First Iteration

Since the data from the pre-design calculations indicated that the filling time of the capillaries was within an acceptable range, design on the simplest form of gel loading mechanism was begun. Since the primary variable influencing the velocity of the gel was the dynamic viscosity μ, the system was designed to accommodate several means of reducing μ, the primary means of which was applying heat.

The other parameters that the gel mechanism had to accommodate were the existing capillary clamping mechanism, the need for gas sealing, modularity in gel-type or DNA loading media, and easy removability from the entire UTMS system. Additionally, to complete the capillary loading testing, the system needed to be operational outside the UTMS.

In order to accurately time the capillary fills independently for each channel, a control program in VisualBasic.NET program was created to work in conjunction with a data acquisition card (DAQ) and a power supply. Because the gel allows a low conductance across a filled capillary, it was used to complete a circuit to trip the virtual stopwatch on each channel in the control program. A screenshot of the control program is shown below in Figure 8.

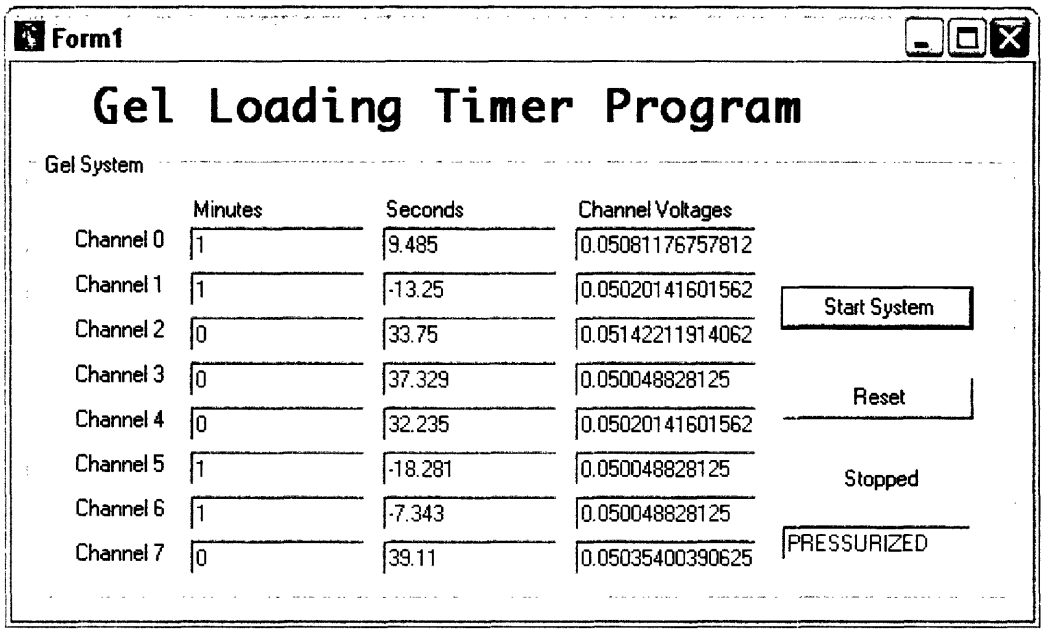

Figure 8: A screenshot of the control program for the gel loading system.

In order to start the virtual stopwatches in the control program at the exact moment the compressed nitrogen was released into the pressure vessel, a solenoid valve was controlled by the program to pressurize the system at the same instant the stopwatches started. The valve was operated by a power supply which was controlled by the program via a GPIB (IEEE 488) interface.

The main body at the bottom was machined from a solid block of aluminum in order to create a pressure vessel of high thermal conductivity that could be attached easily to the bottom of the UTMS main apparatus. In order to manipulate the viscosity of the gel by changing the temperature, an external hot plate was able to be placed below the apparatus and apply heat to the system via open-loop control. A hole drilled just below the surface of the bottom of the pressure vessel allowed a thermocouple to be inserted at a

point where the thin aluminum between the base block and the gel pan was nearly isothermal.

Holes were drilled and tapped into the base block to incorporate the NPT fittings of the pressurized nitrogen tube and the pressure transducer. The top of the block was machined with many holes and fitted with a silicone seal to maintain pressure within the vessel. In the first iteration, each channel was equipped with its own Eppendorf tube in which to place the gel for loading. The capillary would dip into this tube, and under pressure, the gel would be forced into the capillary.

To make a timing mechanism, a circuit arrangement was fabricated that used the moving gel to complete an electrical connection. Several iterations of the circuit were developed en route to making a successful electrical connection through the conductive gel. The first iteration used a wire electrode which was inserted into an Eppendorf tube along with the empty capillary inside the pressure vessel, as shown below in Figure 9. The electrode in the gel acted as the positive side, or the cathode. The free end of the capillary was dipped into a second Eppendorf tube along with a second wire electrode, the anode, and was submerged in a conductive electrolyte buffer. When the gel would squirt out the free end of the capillary into the conductive buffer, the circuit would complete and current would flow across the resistor, across which the DAQ would measure a voltage and trigger the stopwatch for that channel.

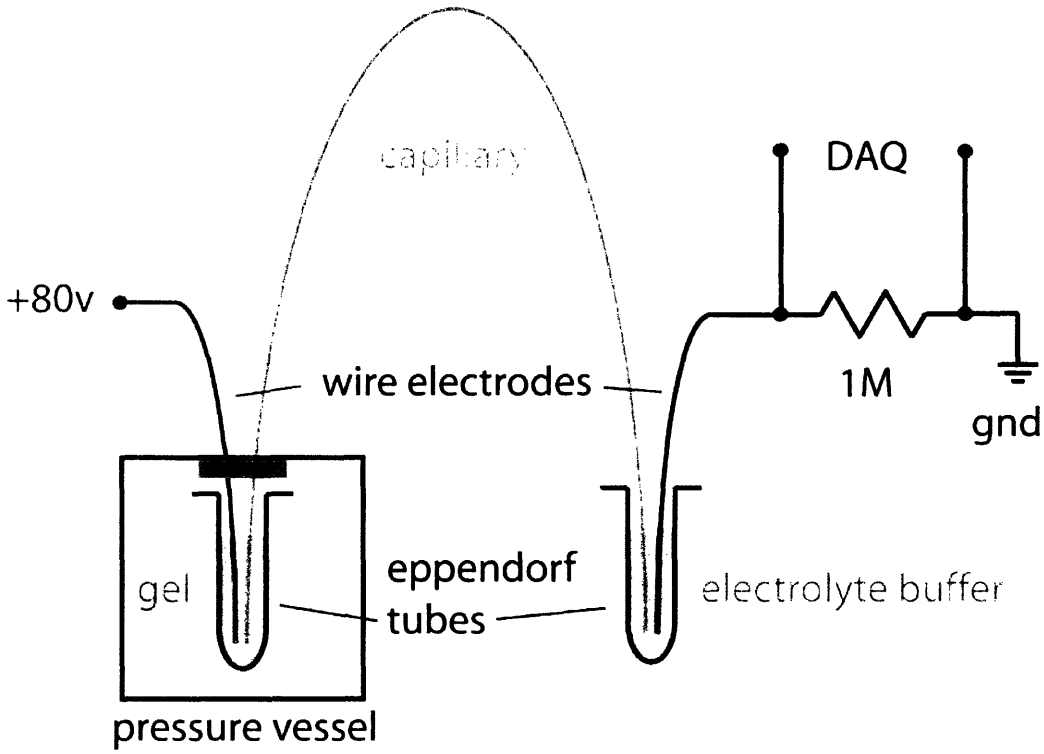

Figure 9: First iteration schematic of gel loader/timer setup with conductive electrolyte buffer.

A 10 channel version of this setup was constructed, and is shown below in Figure 10.

Figure 10: First iteration of multichannel loader, using Eppendorf tubes and electrolyte buffer.

This iteration proved the concept with the pressure vessel and the timing circuit. However, the use was impractical and less pertinent to the final UTMS design for several reasons. The 10,000 channels will be filled from a single bath. At 1mm rectangular spacing for each of the 100 × 100 channels, an individual well plate array for filling gel is

impractical and unnecessary. Therefore, the single Eppendorf tubes for each channel was not indicative of the performance from a common bath.

Additionally, a 5,000 volt potential was used across the electrodes in order to achieve a 5 v drop across the 1 mega-ohm resister which the DAQ used to measure the signal. This extremely high potential caused arcing within the system, as well as conducting electric fields which induced shorts in other areas. Evaporation of the electrolyte buffer was also a significant problem, and it was difficult and tedious to replace the exhausted and dried buffer after each run of the system.

5.2 Design of Second Iteration

The second iteration accounted for all of these problems. First, the impracticality of individual Eppendorf tubes for each channel was eliminated by changing to a common bath, and spacing the capillaries at their designated spacing within the UTMS: 1mm each direction, in a rectangular array. This allowed for a smaller gel volume to be used for loading, and reduced the waste associated with residue in many separate loading containers. A solid model of the second iteration is shown below in Figure 11, and shows the interface between the capillary clamping mechanism, the top plate, seal, pipe fittings, and solenoid valve.

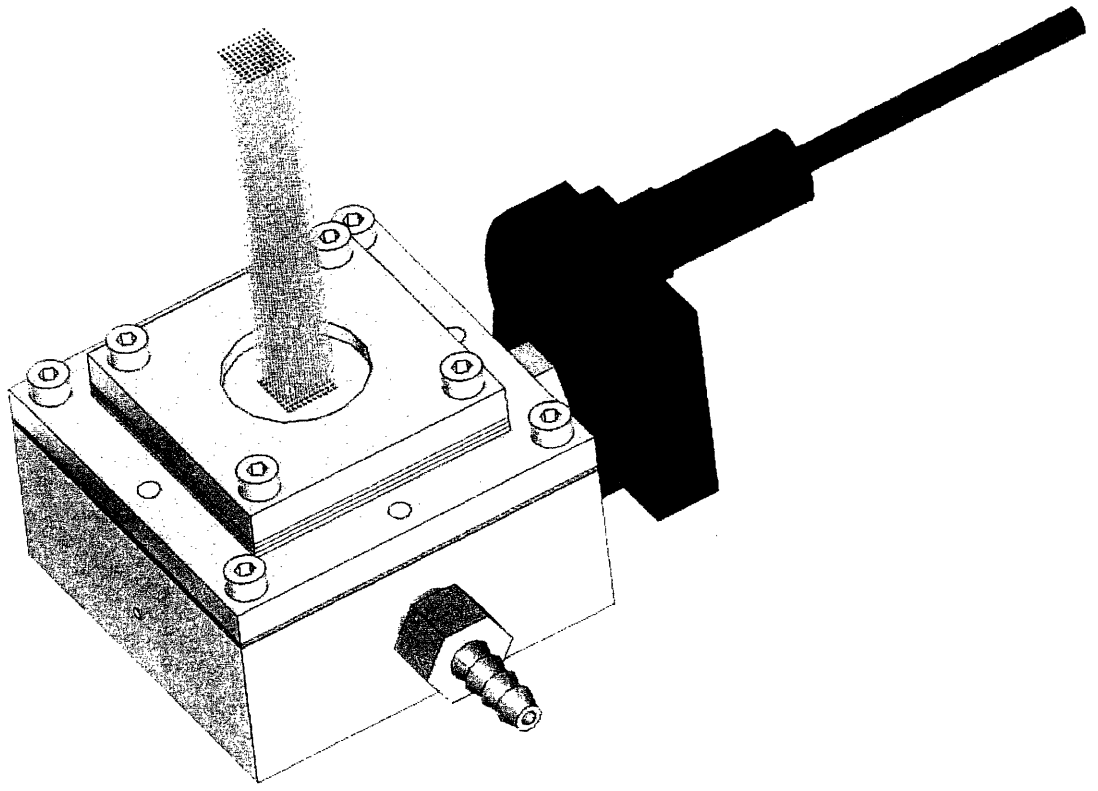

Figure 11: Solid model of the scalable 100 capillary gel loading system.

A more significant and effective redesign was that of the timing mechanism. The gel itself was to be used as a conductor to electrically trigger the stop of a timer. In the previous iteration, as mentioned, a connection was made from the gel to the electrical circuit via a conductive electrolyte fluid which was in contact with both the capillary and the anode. In order to make an electrical connection from the gel to the circuit, the fluid contact was replaced with a tight physical constraint for the capillary which pressed its end against a copper electrode. The schematic for this newer system is shown below in Figure 12.

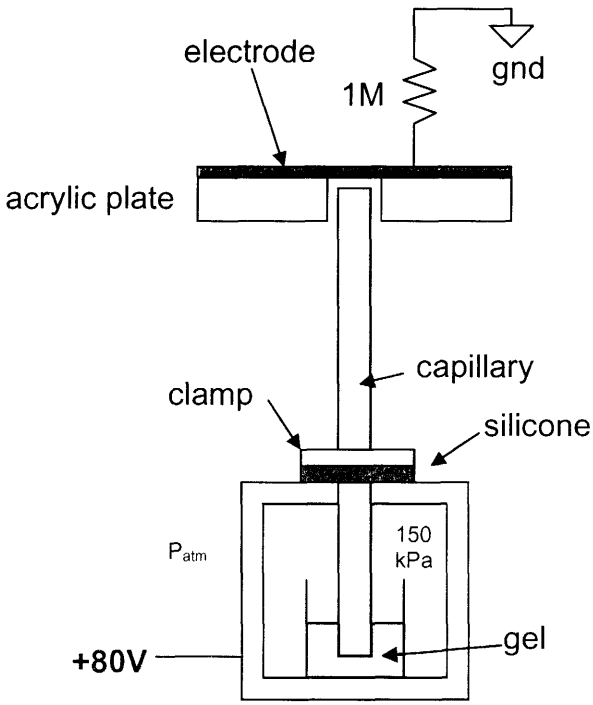

Figure 12: Schematic for second iteration of setup.

This copper electrode was on a circuit board which included the necessary connections to the timing circuit, so that when the gel was ejected at the end of the full capillary, the circuit was complete. The circuit board with the electrodes was constructed to be movable, so that in between runs it could be cleaned of gel residue to ensure a false signal would not occur on the next run. The constraint for the capillaries was constructed from acrylic using a laser cutter. The laser cutter precisely cut holes to align the capillaries to the positions of the electrodes along the circuit board. This constraint and electrode system is shown below in Figure 13.

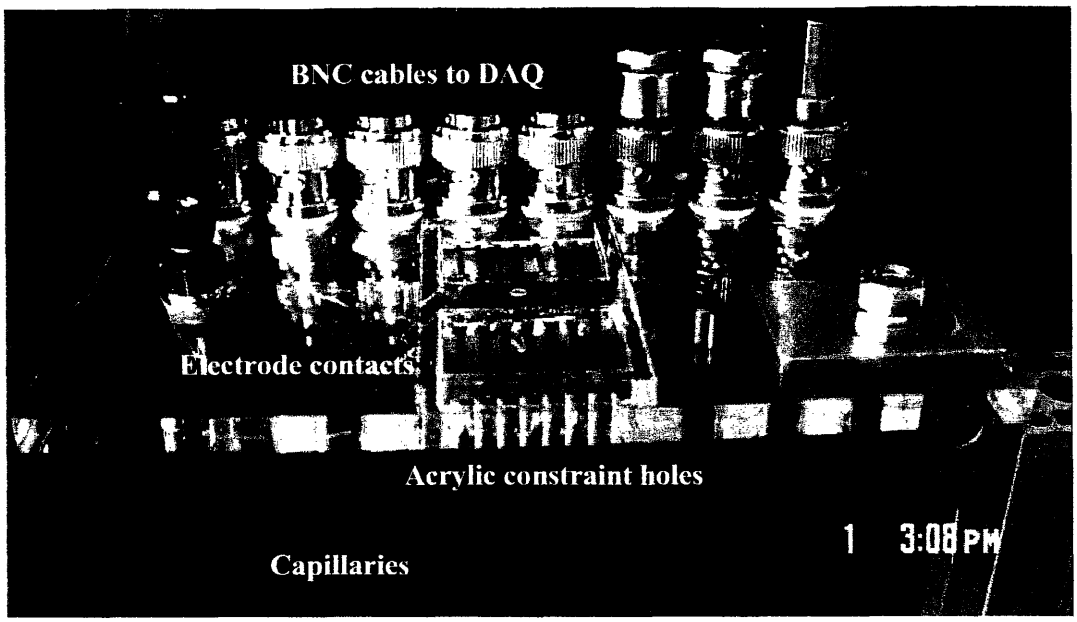

Figure 13: Capillary constraint and electrode system for operation of timing circuit.

The seal around the capillaries for the pressure vessel was created using the clamp mechanism which constrains the array within the temperature control chamber. The clamp was constructed of two plates of stainless steel with hole arrays through them. The plates are clamped together via bolts at each corner, and squeeze three layers of soft silicone. Due to the expansion of the silicone under pressure, a solid mechanical hold is created which grips each capillary individually, and seals the system against significant pressure. The entire clamp assembly is bolted to the pressure vessel and is sealed with yet another layer of soft silicone.

It was important to incorporate this clamp mechanism into the gel loading pressure vessel, since a similar scaled clamp will be used on the 100 × 100 array. The verification of the compatibility of both scaled models assured the functionality when scaled up by 100. Additionally, by keeping the clamp fixed to the optical table, the gel system was

able to be removed, cleaned, the gel replaced, and even capillary wash loaded without disturbing the careful positioning of the capillaries themselves.

Finally, an additional 1/8" NPT hole was tapped to accommodate the input from the solenoid valve. By controlling the power supply for the solenoid valve with the control program, the timer started at the exact moment the pressure was induced. This ensured reliable, automatic timing of the loading system. The second iteration of the gel loading device was used successfully to collect data on the loading of capillaries at different lengths and pressures for 45 total runs. A view of the pressure vessel and capillary clamp is shown below in Figure 13.

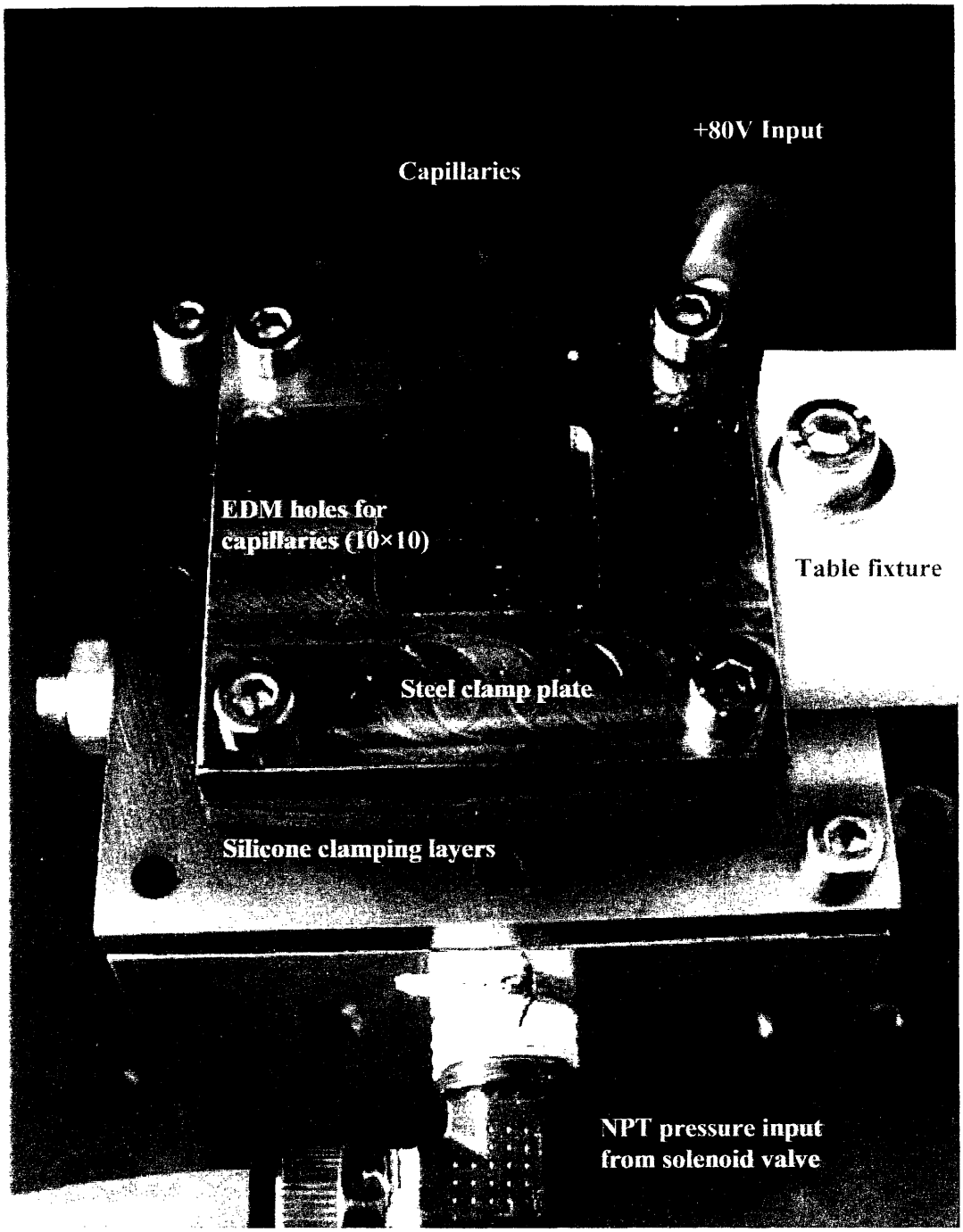

Figure 14: Capillary clamp and pressure vessel.

6.0 Procedure

As can be seen in Figure 14 above, the capillary clamp and top plate of the pressure vessel were fixtured to the optical table, and the pressure vessel itself was suspended from them. Before each run, the tray as shown in the schematic in Figure 9 was filled with capillary wash, and the pressure vessel was mounted to the clamp and plate. Pressure was manually applied via a solenoid valve controlled by the Visual Basic control program, and the capillaries were "washed" until the fluid came out of each capillary at the same constant rate.

Next the system was depressurized, and a new tray was placed in the pressure vessel which contained the gel mixture. The pressure vessel was bolted back to the clamp-plate, and the control program was started. The program then simultaneously opened the solenoid valve and started its timer, scanning the DAQ channels constantly for the prescribed voltage threshold which indicates a filled capillary. At the completion of filling of all the channels, the control program turned off the solenoid valve and depressuried the system, and the fill times of each channel were recorded.

After the gel run, the gel tray was replaced again with the capillary wash tray. The system was again manually repressurized, and was kept pressurized until the capillary wash had pushed all the gel out of every channel and the wash exited all the capillaries at the same constant rate. Once the capillaries were clear, the gel tray was inserted, and the process was repeated.

7.0 Data

Seventy-two total runs were conducted for each set of variables, with each of the 5 sets of variables corresponding to a single data point. The high number of runs was conducted in order to ensure an accurate mean for each data point. Three data points were found in order of increasing pressure at a constant length, and two more taken which corresponded to increasing length at the greatest pressure. This gave 3 data points for both the length and pressure relationships, which was enough to construct an accurate line and equation describing the relationship. The middle point, at 300 mm length and 250 kPa, was used twice.

For the increasing pressure runs, 8 300mm capillaries were timed simultaneously, 9 times in a row, for 72 total data points. This nine-run cycle was completed at 150 kPa, 200 kPa, and 250 kPa. At 250 kPa, 9 runs were taken at additional lengths of 350mm and 400 mm to characterize fill time in relation to capillary length. Below in Figures 15 and 16, the relationships are shown between loading time and pressure, and loading time and capillary length, respectively. The actual values for the run times for pressure and length are listed in Appendix A in tables A1 and A2, respectively.

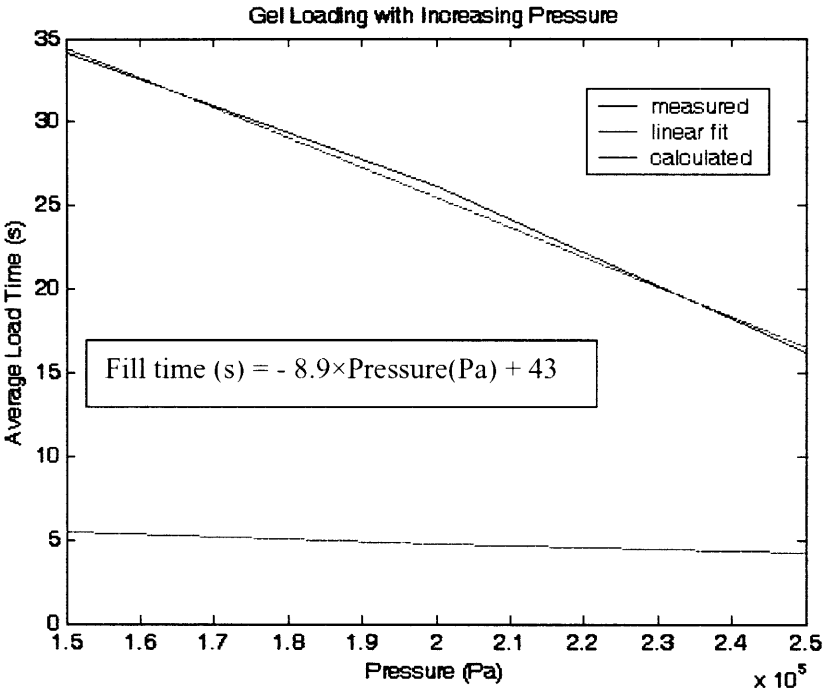

Figure 15: Loading times for constant 300mm length as pressure increases.

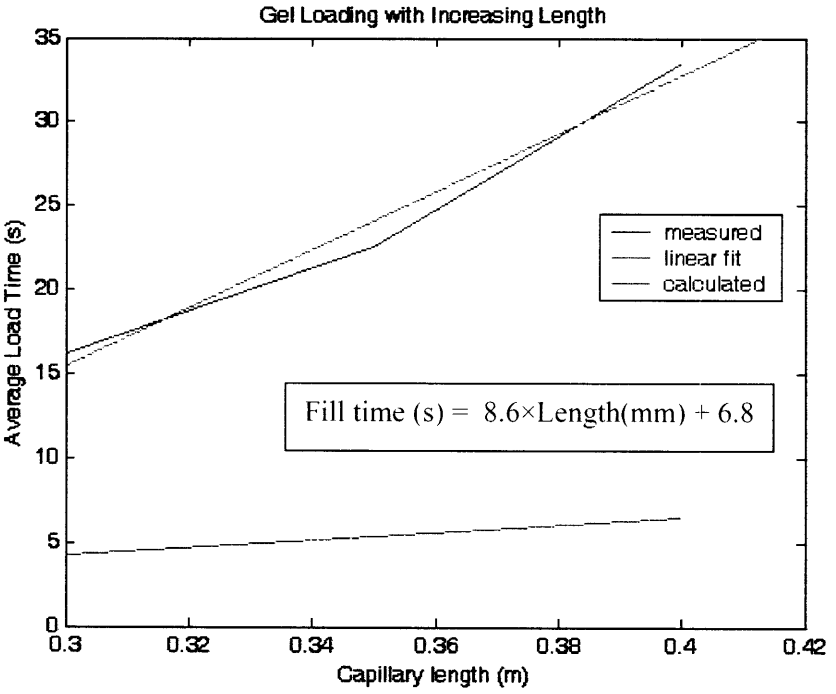

Figure 16: Loading times as capillary length increases with constant pressure at 250kPa.

In order to observe the distributions of fill times for each data point in Figures 15 and 16, histograms were created with 2-second bins. By comparing the mean of each data set in the histograms, the linear relationship between the varied length and pressure and capillary fill time can be observed.

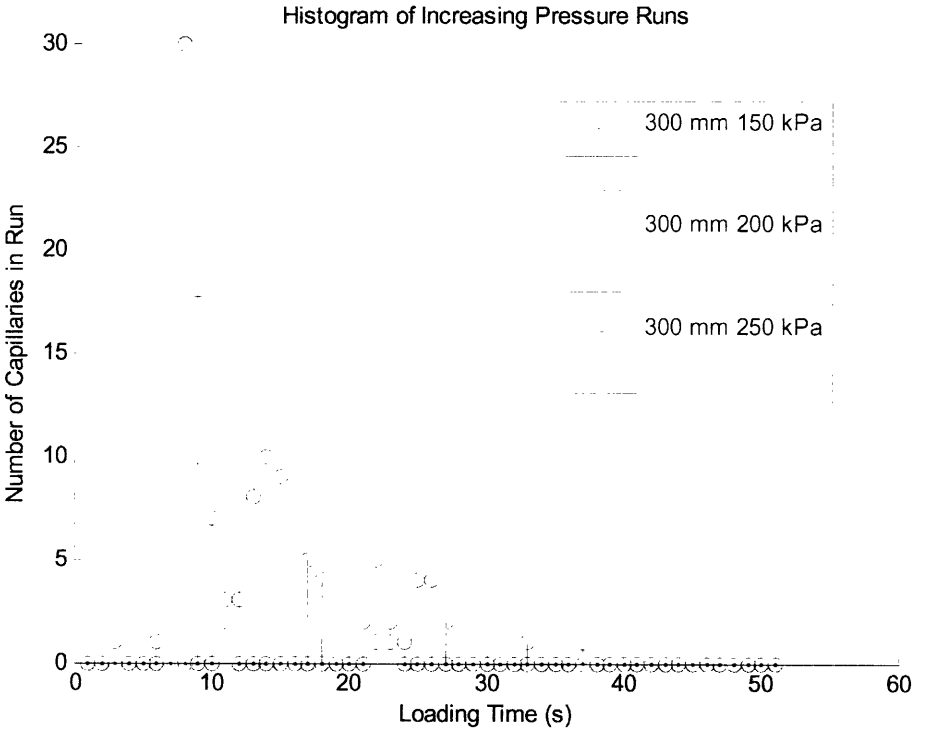

Figure 17: Histogram of runs with increasing pressure.

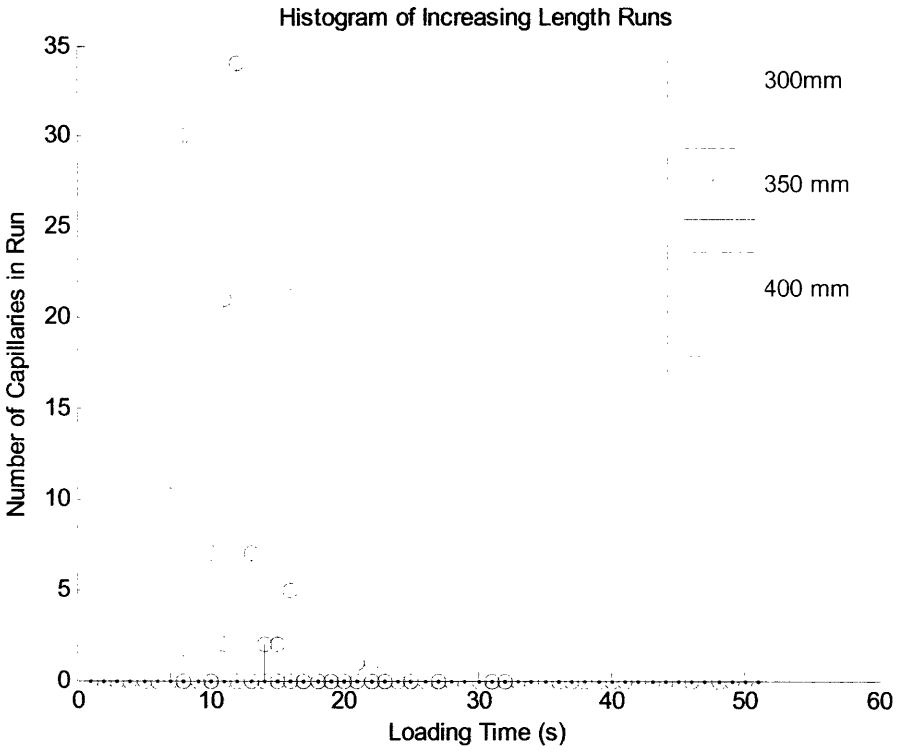

Figure 18: Histogram of runs with increasing length.

Variability between capillaries for each set of injection parameters was compared to observe trends in differing injection times. Average times per capillary for increasing pressure are shown in the appendix in Table A3 below, with increasing length shown in Table A4. The same values are illustrated graphically in the chart in Figures 19 and 20 below.

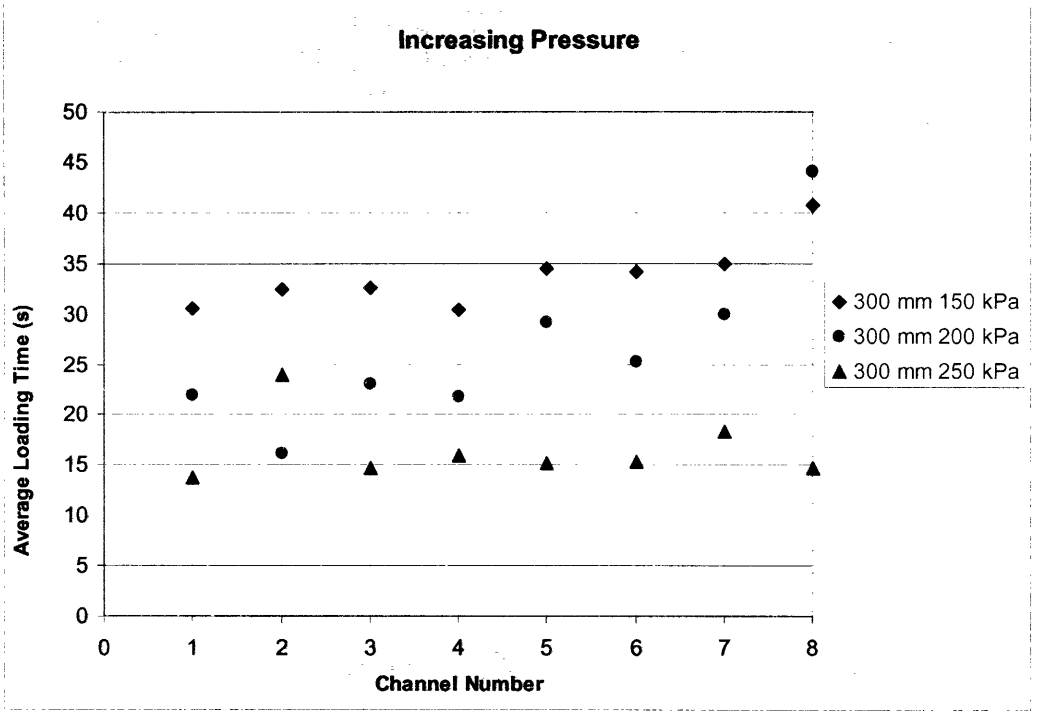

Figure 19: Average injection times per channel for 300mm, for increasing pressure.

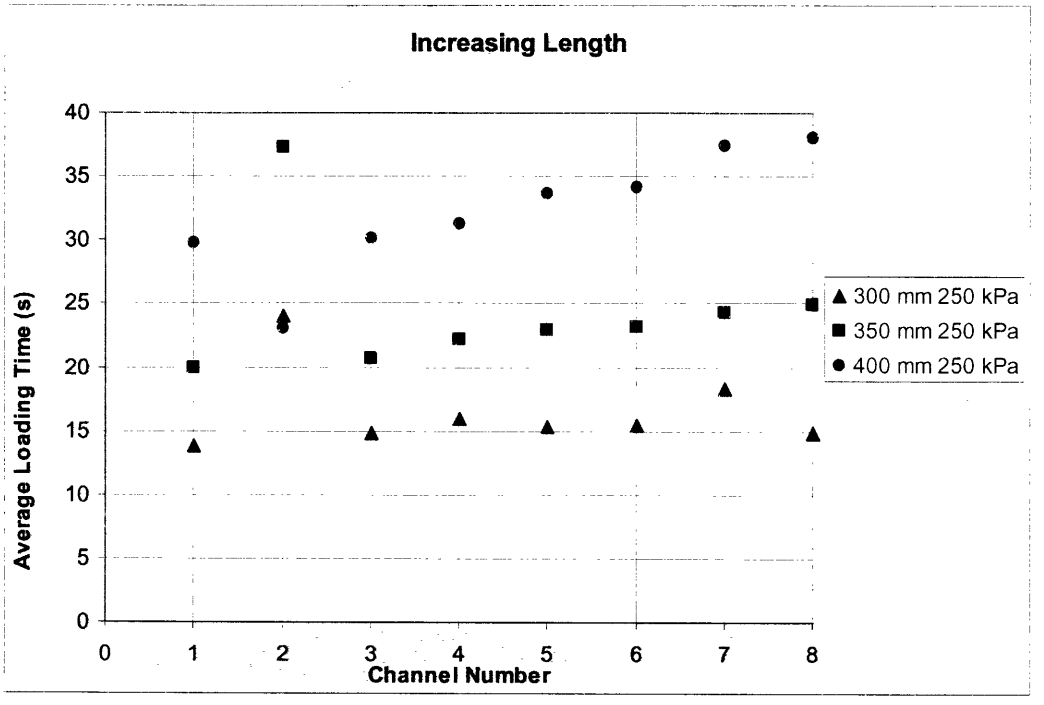

Figure 20: Average injection times per channel at 250kPa, for increasing length.

One-way ANOVA was also calculated for each of the 9 runs in every data point, to see the probability that the mean of each run was close to the actual mean if many more runs were measured. The ANOVA data for each set is listed below in Table 1, and the graphic representation of the means, standard deviations and minimum and maximum values are shown below in Figures 21 and 22.

Table 1: ANalysis Of VAriance (ANOVA) data for each data set.

	Parameters	F-Value	p-value
Increasing Pressure	150 kPa, 200 kPa, 250 kPa	45.92	0
Increasing Length	300 mm, 250 mm, 400 mm	122.52	0

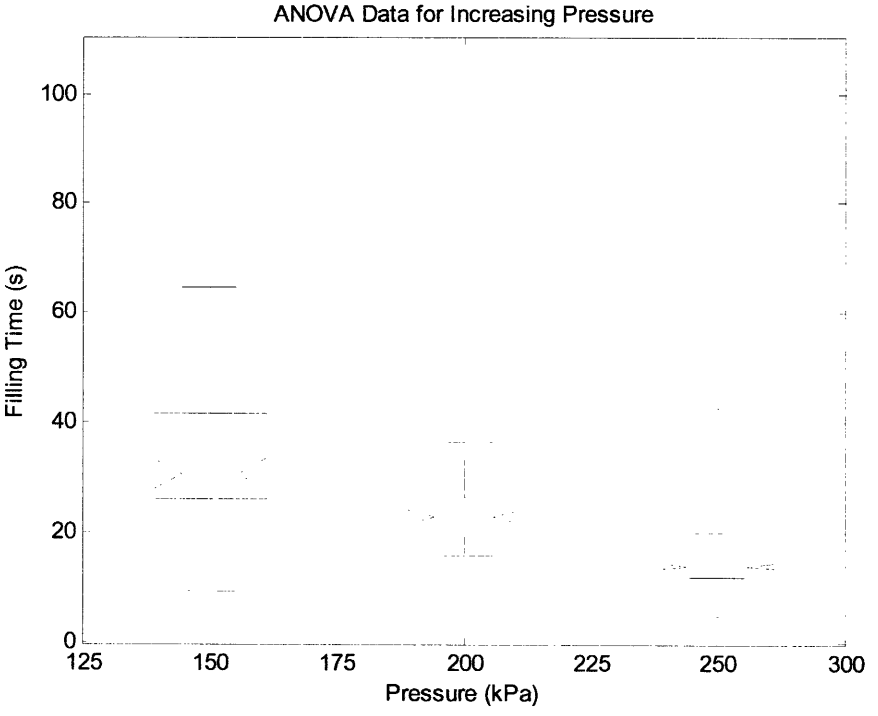

Figure 21: ANOVA data for increasing pressure. Red lines are means, blue is standard deviation, black is max/min value.

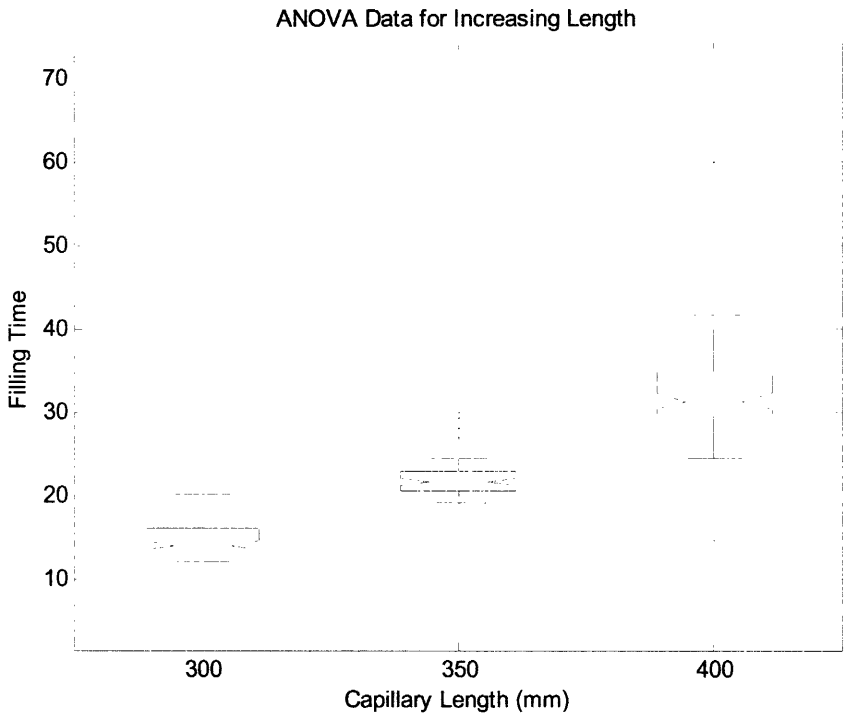

Figure 22: ANOVA data for increasing length. Red lines are means, blue are standard deviation for groups, and black are max/min values.

The F-Value in the ANOVA data is the variance between group means divided by the expected variance within the groups. The p-value is the probability of the null hypothesis, that increasing the pressure or the length has no effect on the mean of the data.

8.0 Discussion and Implications of Data

The data collected by the gel loading instrument totaled 9 runs for each of the 5 parameter sets, with 8 capillaries in each run. This resulted in 360 individual capillary times. In not a single run did any channel fail to load gel all the way through the

capillary; however, there were some trends for loading speed between capillaries which appeared to continue across different parameter sets.

Observing the average times per channel in Figures 19 and 20, it is clear that several channels were consistently slower during operation. In Figure 20, Channel 2 showed significantly slower operation at 300 mm and 400 mm. In Figure 19, for pressures at 150 kPa and 200 kPa, Channel 8 was consistently slow. This trend was noticed mid-data collection. In an attempt to identify the source of the slow performance, after each slow run, the capillary in Channel 8 was switched in position with Channel 1. Immediately after this switch, Channel 8 would still be slower than the rest.

This indicates that the reduced flow velocity is not due to some inhibition in the capillary itself, but is rather due to some flow tendency in the filling container. The slow filling time was also qualitatively observed during the capillary wash process. Channel 8 consistently took longer to wash than the rest of the capillaries. During the runs where Channel 2 was slowest, the same situation was observed: while pressurizing the system with a pan full of capillary wash, which would displace the used polymer gel to ready the capillary for the next run, the displacement of the polymer gel took significantly longer. The washing of the capillaries is described in more detail in Appendix B.

If the problem is not identified and solved, slow capillaries will potentially be a significant problem during loading of the full-scale UTMS. Polymer gel separation matrix is expensive, and due to the tendency of the fluid to go down the capillaries with

the least fluid resistance, i.e. the capillaries which have already completed loading and are not jammed, high amounts of gel waste potentially exist during loading, if the UTMS operator waits for the slow channels to fill.

One possible solution for this is to place a fluid resistor at the end of each capillary. This fluid resistor could take form simply as a reduced diameter which would inhibit gel flow at the end of the capillary, but would let the air in the capillary easily escape. Then, when the gel reached the end of the capillary, that channel would reduce its flow rate, and pressure would be diverted to capillaries which are not yet full. A schematic of such a fluid resistor is shown below in Figure 21.

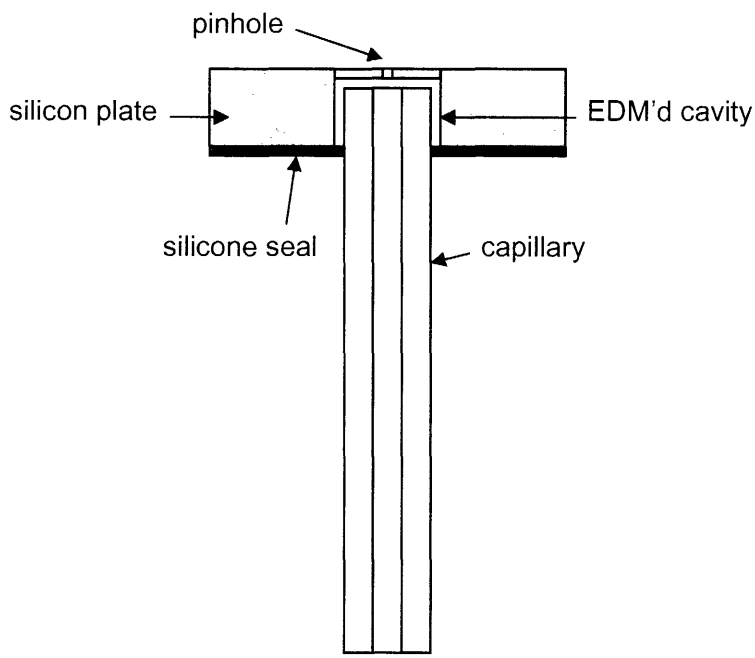

Figure 23: Flow inhibitor for evening of capillary array filling.

By using a sink EDM (electron discharge machining apparatus) to remove material in a silicon plate which would leave room for the capillary tip with a pinhole above for airflow, a flow inhibitor could be created to even the gel flow between the faster and slower capillaries. Using a soft silicone seal as on the buffer well array in Figure 4, a seal could be created which would contain a small volume of gel within the flow inhibitor, preventing unwanted gel waste during filling.

Gauging the effect of varied pressure and varied length in changing the fill times for the capillaries, the expected relationship calculated in the theory section was verified. Increased pressure certainly does decrease fill times, just as increased length slows the process. Interestingly, observing the variation of the standard deviation between parameter sets, it is clear that some sets appear to have more consistent operation than others. Increasing the pressure seemed to reduce the standard deviation, as did increasing the length. This trend is intuitively reasonable, since increased pressure would make a given flow inhibitor, such as a piece of dust, less of an effect on the viscous flow. Additionally, for a longer length, the standard deviation should reduce for a given flow inhibitor, since the decreased time from that inhibition is then a smaller fraction of the total time.

If the variability in fill time is to be significantly decreased (and the fill time itself decreased as well) it is the clear that the pressure for the system should be increased. Because of the impracticality of the setup, temperature variation was not tested in the machine. The likely result of increasing the temperature of the system, however, is also

decreased fill times and decreased variability, due to the reduction in viscosity of the polymer gel. At a lower viscosity, a given flow inhibitor will have less of an effect, as will the viscous shear with the capillary walls. Since the entire UTMS system will be temperature controlled around the 80 °C range, it is certainly conceivable to manually lower the viscosity of the gel during filling to increase the fill performance. Calculations from the theory section with lowered viscosity indicate the proportional decrease in fill time.

The ANOVA data at the end of section 7.0 shows statistically that by varying the parameters of length and pressure, the data is indeed affected. Though this is the expected outcome, it is a positive verification that the by varying the intended aspects of the system, the null hypothesis was correct. Because the p values were both below 0.05, the independent variables (length and pressure) were indeed significant, as predicted by the theory, and by common sense.

From the linear fit lines in Figures 15 and 16, equations were found which can be used to predict the fill time based on the length and pressure, and are shown below in Equations 13 and 14 below.

$$\text{Fill time (s)} = -8.9 \times \text{Pressure(Pa)} + 43, \qquad \textbf{13}$$

where in Equation 13, the capillary length is 300mm. Equation 14 gives the relationship for fill time versus length, at a constant pressure of 250kPa.

$$\text{Fill time (s)} = 8.6 \times \text{Length(mm)} + 6.8 \qquad \textbf{14}$$

9.0 Conclusion and Further Testing

With the completion and successful testing of the multichannel polymer gel loader from this thesis, it is clear that scalable, parallel gel loading is highly functional for a large system with thousands of channels, such as the UTMS. The fluid model did not predict numbers for the fill time that were close to what was measured, but the trend was the same, and it showed that the linear relationship between the pressure, length, and fill time that was measured by the data was indeed accurate.

The testing capabilities enabled by this reliable device go beyond the tests performed for this thesis. The system was build with aluminum in order to allow for complete isothermal operation, so that temperature tests could be performed for testing reduced viscosity run times. Additionally, the plate can accommodate 92 more capillaries. By sensing the current across each channel on the DAQ board instead of simply the voltage threshold, multiple capillaries could be run on the same channel, allowing for further scaled up testing before the full 10,000 capillaries. With the additional data, more accurate measurement of the effect of various parameters on standard deviations can be further explored, as well as the trends between capillary placement and fill time. Not enough data was taken while trying to focus on these particular aspects of the filling, and it may become more important if the effects limit the performance of a larger device.

Incorporation of a full 100 × 100 fill array into the bottom of the UTMS temperature chamber will be simple, and will easily accommodate the 3 layer silicone capillary clamp which holds the channels in place. If it is necessary to physically sense the arrival of gel

in each of the 10,000 channels during filling as well, the simplified buffer well plate created to capture the gel and trigger the timer is an effective and reliable method to do so, if the statistical calculations from this thesis are less exact than necessary. By multiplexing and electrically scanning each channel through time, it is a simple thing to physically verify the presence of gel in each of the 10,000 capillaries.

Lastly, in order to test if the variability in fill time can be reduced using a flow inhibitor array, it would be very useful to construct a prototype using layered acrylic and test it with the system. If it is found that it significantly reduces the standard deviation in fill times, it would be worthwhile to incorporate into the full array in order to reduce gel waste during filling.

Together, the design of the gel loader and the testing and subsequent analysis of its performance push forward the ultra-high throughput mutational spectrometer, and bring it one step closer to fruition. The system is highly functional and reliable, and has proven its effectiveness for the full UTMS, as well as introduced possible design solutions for even further improvement. Finally, the equations found by the data that was taken are useful in determining the fill time based on the pressure and length.

References:

[1] Cotton, R.G.H. *Mutation Detection*: Oxford University Press, 1997.

[2] Munson, B.R. Young, D. F. Okiishi, T. H. *Fundamentals of Fluid Mechanics*, 3rd Edition: John Wiley & Sons, 1998.

[3] White, F. M. *Fluid Mechanics*. McGraw-Hill, 2003.

[4] Scherer, J. R. Paegel, B.M. Wedemayer, G.J. Emrich, C.A. Lo, J. Medintz, I.L. Mathies, R.A. *High-pressure gel loader for capillary array electrophoresis Microchannel Plates*. BioTechniques 31:1150-1154 November 2001.

[5] Khrapko, K. Coller, H. Andre, P. Li, X. Foret, F. Belenky. A. Karger, L. Thilly, W. *Mutational spectrometry without phenotypic selection: human mitochondrial DNA*. Nucleic Acids Research, 1997, Vol 25, No. 4. 685-693.

[6] Li-Sucholeiki, X. Khrapko, K. Andre, P. Marcelino, L. Karger, B. Thilly, W. *Applications of constant denaturant capillary electrophoresis/high-fidelity polymerase chain reaction to human genetic analysis*. Electrophoresis 1999, 20, 1224-1232.

[7] Khrapko, K. Hanekamp, J. Thilly, W. Belenkii, A. Foret, F. Karger, B. *Constant denaturant capillary electrophoresis (CDCE): a high resolution approach to mutational analysis*. Nucleic Acids Research, 1994, Vol 22, No. 3, 364-369

[8] http://www.physics.csbsju.edu/stats/anova.html

[9] Freeman, S. *Biological Science*, Prentice Hall, 2002.

[10] http://www.eastman.ucl.ac.uk/research/MD/ecology_community_analysis.html

[11] http://microcribi.cribi.unipd.it/agilent_interpreta.htm

[12] Weekly BioInstrumentation Lab Meeting Presentation (04/08/2005) by C.R. Forest

[13] Internal BioInstrumentation Lab UTMS reference, by I.W. Hunter

[14] Kreyszig, E. *Advanced Engineering Mathematics, 6th ed.* John Wiley & Sons, Inc, 1988.

[15] Montgomery, D.C. Runger, G.C. Hubele, N. F. *Engineering Statistics*. John Wiley & Sons, Inc. 1998.

Appendix A: Detailed data representations

Table A1: Calculated and measured injection times with increasing pressure (corresponding to Figure 15)

Parameter Set	Parameters At 26 °C	Calculated Time (s)	Measured Time (s)	St.Dev. (s)
1	300 mm, 150 kPa	5.458	34.106	11.435
2	300 mm, 200 kPa	4.725	26.107	13.250
3	300 mm, 250 kPa	4.226	16.231	8.414

Table A2: Calculated and measured injection times with increasing length (corresponding to Figure 16)

Parameter Set	Parameters At 26 °C	Calculated Time (s)	Measured Time (s)	St.Dev. measured (s)
3	300 mm, 250 kPa	4.226	16.231	8.414
4	350 mm, 250 kPa	5.324	22.608	3.411
5	400 mm, 250 kPa	6.503	33.530	7.245

Table A3: Average injection times per channel, for increasing pressure (corresponding to Figure 17).

Length = 300 mm	150 kPa	200 kPa	250 kPa
Channel 1	30.574 s	22.000 s	13.827 s
Channel 2	32.396 s	16.068 s	24.009 s
Channel 3	32.609 s	23.017 s	14.785 s
Channel 4	30.349 s	21.714 s	16.007 s
Channel 5	34.513 s	29.090 s	15.275 s
Channel 6	34.152 s	25.204 s	15.435 s
Channel 7	35.007 s	29.995 s	18.377 s
Channel 8	40.806 s	44.017 s	14.783 s

Table A4: Average injection times per channel, for increasing length (corresponding to Figure 18)

Pressure = 250 kPa	300 mm	350 mm	400 mm
Channel 1	13.827 s	19.968 s	29.740 s
Channel 2	24.009 s	37.267 s	23.000 s
Channel 3	14.785 s	20.722 s	30.195 s
Channel 4	16.007 s	22.228 s	31.334 s
Channel 5	15.275 s	22.972 s	33.631 s
Channel 6	15.435 s	23.124 s	34.160 s
Channel 7	18.377 s	24.259 s	37.351 s
Channel 8	14.783 s	24.914 s	37.959 s

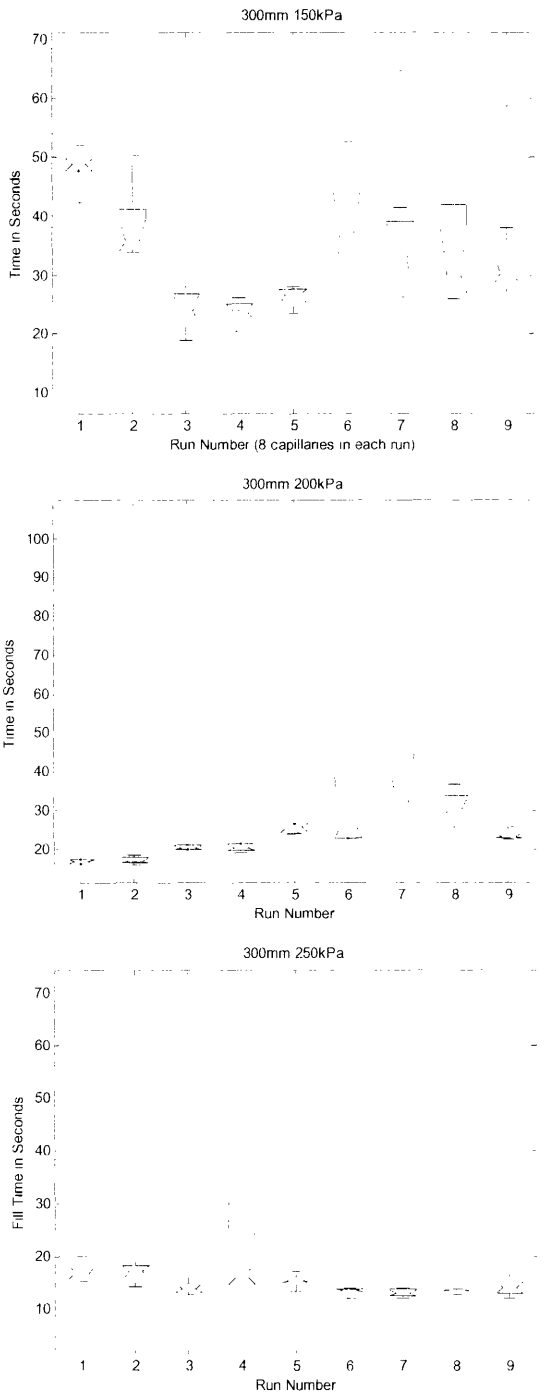

Figure 24: ANOVA Data of each set with increasing pressure. Red is the mean, blue the standard deviation, and black the minimum and maximum values.

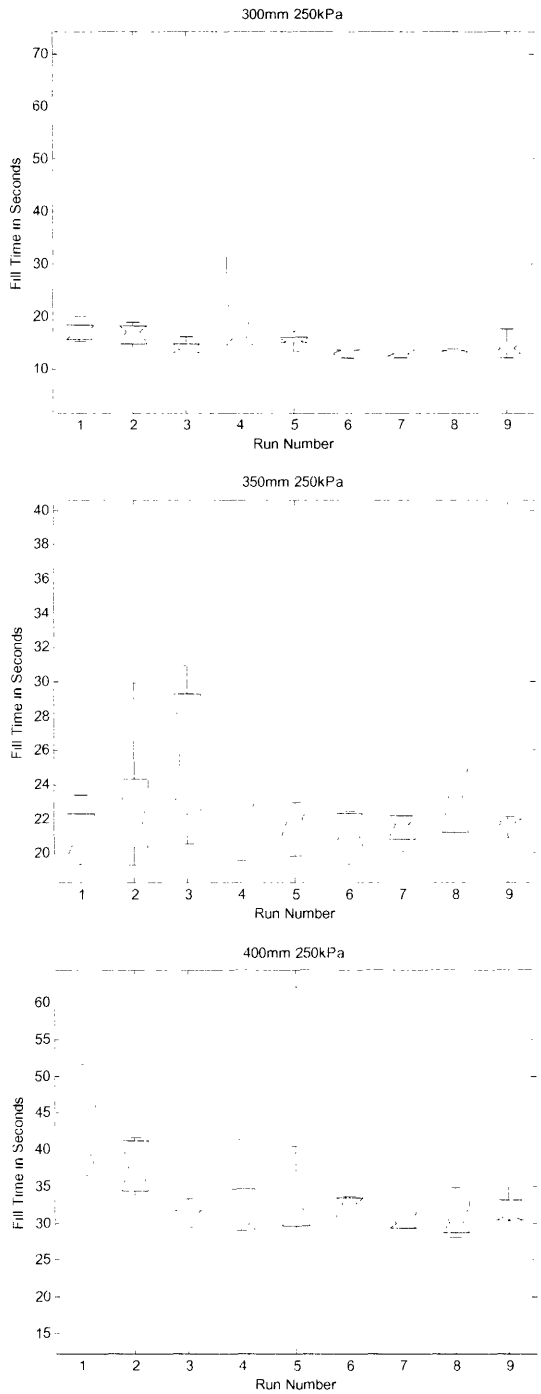

Figure 25: ANOVA Data for each set with increasing length runs. Red is the mean, blue the standard deviation, and black the minimum and maximum values.

Appendix B: Capillary Washing

Another important task qualitatively accomplished by the test runs performed in this experiment was the characterization of the wash time for the capillary system. As described in the Procedure section (6.0), it is imperative to change the gel in every capillary in between DNA runs. Evacuating the used gel is accomplished by replacing the gel pan in the pressure vessel with a pan filled with capillary wash. Capillary wash is then injected through the channels until the gel has been displaced from all of them, and the wash is exiting each capillary at the same rate.

The time required to wash the capillaries was hand-timed in order to qualitatively characterize the washing process. It was found that in nearly ever case, if the system was pressurized for the same amount of time as it took to load the set of capillaries, all but the "slow" capillary would be completely washed. It consistently took approximately 10 additional seconds for the slow capillary to be completely displaced of polymer gel.

The washing process exhibited the problem with parallel capillary loading discussed in section 8.0, but more strongly. As related in section 8.0, when a capillary exhibits a lower fluid resistance than the others, the flow rate through that capillary will be higher, and the flow through the higher resistance channels will be lower. Thus, if one capillary is slow, it will become even slower if it still has gel to displace while the others have a full flow of capillary wash.

The different washing times due to varied fluid resistance is a difficult problem to alleviate, since the fluid resistor solution proposed in section 8.0 would not work. A fluid resistor system would be ineffective since the gel being displaced is of a higher viscosity than the capillary wash, meaning the resistor would have to be added after the gel had been displaced. Additionally, an issue that needs to be addressed is the disposal of the capillary wash that comes out of the high-flow capillaries while the system displaces the "slow" channels. A significant volume of capillary wash is ejected during the washing process, and this volume will have to be disposed of properly in order to not interfere with the rest of the instrument.

One suggestion is to place a large sponge over the tips of the capillaries. This sponge would both catch the used polymer gel, as well as absorb the extra capillary wash. Upon displacement and refilling of the gel, the sponge could be removed, the micro-lenses replaced, and the system restarted on another run.

Appendix C: MATLAB code for data manipulation

1: CAPMAIN. Primary program.

```
%Main capillary data analysis program. Utilizes function alter x' to
%correct for negative fill times by adding 60 to values which are
%negative. Corrects data for all parameter sets, labeled in matrices by
%P1, P2, etc.

%P's rows are capillaries
%cols are runs
%ind=x  ind dim  are independent variable, different data sets
%pressure, length, etc

clc;
numfiles=5;

%loads data files, where raw data is stored in 8x8 arrays
for m=1:numfiles
    filename=strcat('P',num2str(m),'.dat');
    a=load (filename);
    P(:,:,m)=a;
end

[x,y,z]=size(P);

%converts negative times stored from vb.NET to positive times
PAconverted = alter(P);

%converts each dataset layer of full data array  8x8x5  into 5 separate
%8x1 arrays
param1 = histo(PAconverted(:,:,1));
param2 = histo(PAconverted(:,:,2));
param3 = histo(PAconverted(:,:,3));
param4 = histo(PAconverted(:,:,4));
param5 = histo(PAconverted(:,:,5));

%averages each channel for each run, creates array
for n = 1:numfiles
      chanaves(n,:) = channels(PAconverted(:,:,n));
end

bins = 0:2:100

%executes model program to insert calculated values to plot with
%measured values
model

%mean times for increasing pressure
means = [mean(param1) mean(param2) mean(param3)];
%mean times for increasing length
means2 = [mean(param3) mean(param4) mean(param5)];
```

```
%plots increasing pressure data
Figure(1)
plot(dp,means)
xlabel('pressure (Pa)')
ylabel('average load time (s)')
title('gel loading with increasing pressure')
hold on
plot(dp,timepres)

%plots increasing length data
Figure(2)
plot(L,means2)
xlabel('capillary length (m)')
ylabel('average load time in (s)')
title('gel loading with increasing length')
hold on
plot(L,timeleng)

%creates histogram with 1-second bins of each data set
hist1 = hist(param1,bins);
hist2 = hist(param2,bins);
hist3 = hist(param3,bins);
hist4 = hist(param4,bins);
hist5 = hist(param5,bins);

%standard deviations of each data set
stddevs = [std(param1) std(param2) std(param3) std(param4)
std(param5)];
```

2: MODEL: MATLAB code calculating fill time data from section 4.0 Theory

```
%MODEL Program;
%velocity profile in the glass capillary in an upward orientation.
%by Mats Ball for undergraduate thesis 5-1-05

clc;

R = 0.000075 ;      %m
D = 2*R      ;      %m
mu =30       ;      %ga*s
dp = 150000  ;      %pa
l = 0.3      ;      %m
rho = 1000   ;      %kg/m^3
gz = 9.8     ;      %m/s^2
V = 0.01     ;      %m/s
kl = 0.8     ;      %head loss coefficient
g = 9.8      ;      %m/s^2

Re = rho*V*D/mu;
f = 64/Re;

dp = [150000 200000 250000];
L = [0.3 0.35 0.4];
```

```
%varied pressure calculations for velocity. Constant length.
for i = 1:3
    for dla = 1:300;
        vpres(dla,i) = sqrt((dp(1,i) -
rho*L(1,1))/((kl/(2*g))+f*(dla*.001/D)));
    end
end

timepres = [0 0 0];

%time to fill calculation for pressure varying data
for j = 1:3
    for m = 1:length(vpres);
        timepres(1,j) = timepres(1,j) + 0.001/(vpres(m,j));
    end
end

%varied length calculations for velocity
for k = 1:3;
    for dla = 1:(L(1,k)*1000);
        vleng(dla,k) = sqrt((dp(1,3) -
rho*L(1,k))/((kl/(2*g))+f*(dla*.001/D)));
    end
end

timeleng = [0 0 0];
%time to fill calculation for length varying data

for m = 1:300;
    timeleng(1,1) = timeleng(1,1) + 0.001/(vleng(m,1));
end
for p = 1:350
    timeleng(1,2) = timeleng(1,2) + 0.001/(vleng(p,2));
end
for o = 1:400;
    timeleng(1,3) = timeleng(1,3) + 0.001/(vleng(o,3));
end
```

3: ALTER function to convert negative times in raw data

```
%Alter function: converts negative time values to positive ones

function output = alter(x)

[r,c,d]=size(x);
output=ones(r,c,d);

for k = 1:d;
    for i = 1:r;
        for j = 1:c;
            if x(i,j,k) < 0;
                output(i,j,k) = x(i,j,k) + 60;
            else output(i,j,k) = x(i,j,k);
            end
        end
    end
end
```

4: HISTO function

```
%rearranges data matrix from single parameter set into vector which is
% then plotted in a histogram

function output2 = histo(H)

[r,c]=size(H);
output2 = ones(r*c,1);

q = 0;
p = 0;
n = 1;

for q = 1:r
    for p = 1:c
        output2(n,1)=H(q,p);
        n = n + 1;
    end
end
```

5: CHANNELS function

```
%channels: separates matrix PAconverted into average load times per
% channel per parameter set. by Nate Fall for undergraduate thesis
%5 1 08

function output = channels(x)

[r,c] = size(x);

output = ones(1,c);

for i = 1:c
    output(1,i)= mean(x(:,i));
end
```

Appendix D: Visual Basic.NET code for timing system control program

```
Imports Agilent.TMFramework
Imports System.Threading
Public Class Form1
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents txtbx1 As System.Windows.Forms.TextBox
    Friend WithEvents txtbx2 As System.Windows.Forms.TextBox
    Friend WithEvents txtbx3 As System.Windows.Forms.TextBox
    Friend WithEvents txtbx4 As System.Windows.Forms.TextBox
    Friend WithEvents txtbx5 As System.Windows.Forms.TextBox
    Friend WithEvents txtbx6 As System.Windows.Forms.TextBox
    Friend WithEvents txtbx7 As System.Windows.Forms.TextBox
    Friend WithEvents txtbx0 As System.Windows.Forms.TextBox
    Friend WithEvents btnStart As System.Windows.Forms.Button
    Friend WithEvents Label9 As System.Windows.Forms.Label
    Friend WithEvents btnReset As System.Windows.Forms.Button
    Friend WithEvents Label11 As System.Windows.Forms.Label
    Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
    Friend WithEvents lbl1 As System.Windows.Forms.Label
    Friend WithEvents lbl2 As System.Windows.Forms.Label
    Friend WithEvents lbl3 As System.Windows.Forms.Label
    Friend WithEvents lbl4 As System.Windows.Forms.Label
    Friend WithEvents lbl5 As System.Windows.Forms.Label
    Friend WithEvents lbl6 As System.Windows.Forms.Label
    Friend WithEvents lbl7 As System.Windows.Forms.Label
    Friend WithEvents lbl0 As System.Windows.Forms.Label
    Friend WithEvents txtVolt0 As System.Windows.Forms.TextBox
    Friend WithEvents txtVolt1 As System.Windows.Forms.TextBox
    Friend WithEvents txtVolt2 As System.Windows.Forms.TextBox
    Friend WithEvents txtVolt3 As System.Windows.Forms.TextBox
    Friend WithEvents txtVolt4 As System.Windows.Forms.TextBox
    Friend WithEvents txtVolt5 As System.Windows.Forms.TextBox
    Friend WithEvents txtVolt6 As System.Windows.Forms.TextBox
    Friend WithEvents txtVolt7 As System.Windows.Forms.TextBox
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents txtMin0 As System.Windows.Forms.TextBox
    Friend WithEvents txtMin1 As System.Windows.Forms.TextBox
    Friend WithEvents txtMin2 As System.Windows.Forms.TextBox
    Friend WithEvents txtMin3 As System.Windows.Forms.TextBox
    Friend WithEvents txtMin4 As System.Windows.Forms.TextBox
```

```
    Friend WithEvents txtMin5 As System.Windows.Forms.TextBox
    Friend WithEvents txtMin6 As System.Windows.Forms.TextBox
    Friend WithEvents txtMin7 As System.Windows.Forms.TextBox
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents btnInit As System.Windows.Forms.Button
    Friend WithEvents lblPressure As System.Windows.Forms.Label
    Friend WithEvents btnDepress As System.Windows.Forms.Button
    Friend WithEvents btnManual As System.Windows.Forms.Button
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(Form1))
        Me.txtbx1 = New System.Windows.Forms.TextBox
        Me.txtbx2 = New System.Windows.Forms.TextBox
        Me.txtbx3 = New System.Windows.Forms.TextBox
        Me.txtbx4 = New System.Windows.Forms.TextBox
        Me.txtbx5 = New System.Windows.Forms.TextBox
        Me.txtbx6 = New System.Windows.Forms.TextBox
        Me.txtbx7 = New System.Windows.Forms.TextBox
        Me.lbl1 = New System.Windows.Forms.Label
        Me.lbl2 = New System.Windows.Forms.Label
        Me.lbl3 = New System.Windows.Forms.Label
        Me.lbl4 = New System.Windows.Forms.Label
        Me.lbl5 = New System.Windows.Forms.Label
        Me.lbl6 = New System.Windows.Forms.Label
        Me.lbl7 = New System.Windows.Forms.Label
        Me.txtbx0 = New System.Windows.Forms.TextBox
        Me.lbl0 = New System.Windows.Forms.Label
        Me.btnStart = New System.Windows.Forms.Button
        Me.Label9 = New System.Windows.Forms.Label
        Me.btnReset = New System.Windows.Forms.Button
        Me.Label11 = New System.Windows.Forms.Label
        Me.GroupBox1 = New System.Windows.Forms.GroupBox
        Me.lblPressure = New System.Windows.Forms.Label
        Me.btnInit = New System.Windows.Forms.Button
        Me.Label2 = New System.Windows.Forms.Label
        Me.txtMin7 = New System.Windows.Forms.TextBox
        Me.txtMin6 = New System.Windows.Forms.TextBox
        Me.txtMin5 = New System.Windows.Forms.TextBox
        Me.txtMin4 = New System.Windows.Forms.TextBox
        Me.txtMin3 = New System.Windows.Forms.TextBox
        Me.txtMin2 = New System.Windows.Forms.TextBox
        Me.txtMin1 = New System.Windows.Forms.TextBox
        Me.txtMin0 = New System.Windows.Forms.TextBox
        Me.Label1 = New System.Windows.Forms.Label
        Me.txtVolt7 = New System.Windows.Forms.TextBox
        Me.txtVolt6 = New System.Windows.Forms.TextBox
        Me.txtVolt5 = New System.Windows.Forms.TextBox
        Me.txtVolt4 = New System.Windows.Forms.TextBox
        Me.txtVolt3 = New System.Windows.Forms.TextBox
        Me.txtVolt2 = New System.Windows.Forms.TextBox
        Me.txtVolt1 = New System.Windows.Forms.TextBox
        Me.txtVolt0 = New System.Windows.Forms.TextBox
        Me.btnDepress = New System.Windows.Forms.Button
        Me.btnManual = New System.Windows.Forms.Button
        Me.GroupBox1.SuspendLayout()
        Me.SuspendLayout()
        '
        'txtbx1
        '
        Me.txtbx1.Location = New System.Drawing.Point(192, 64)
        Me.txtbx1.Name = "txtbx1"
        Me.txtbx1.Size = New System.Drawing.Size(96, 20)
        Me.txtbx1.TabIndex = 0
        Me.txtbx1.Text = ""
        '
        'txtbx2
        '
        Me.txtbx2.Location = New System.Drawing.Point(192, 88)
        Me.txtbx2.Name = "txtbx2"
        Me.txtbx2.Size = New System.Drawing.Size(96, 20)
        Me.txtbx2.TabIndex = 1
```

```
Me.txtbx2.Text = ""
'
'txtbx3
'
Me.txtbx3.Location = New System.Drawing.Point(192, 112)
Me.txtbx3.Name = "txtbx3"
Me.txtbx3.Size = New System.Drawing.Size(96, 20)
Me.txtbx3.TabIndex = 2
Me.txtbx3.Text = ""
'
'txtbx4
'
Me.txtbx4.Location = New System.Drawing.Point(192, 136)
Me.txtbx4.Name = "txtbx4"
Me.txtbx4.Size = New System.Drawing.Size(96, 20)
Me.txtbx4.TabIndex = 3
Me.txtbx4.Text = ""
'
'txtbx5
'
Me.txtbx5.Location = New System.Drawing.Point(192, 160)
Me.txtbx5.Name = "txtbx5"
Me.txtbx5.Size = New System.Drawing.Size(96, 20)
Me.txtbx5.TabIndex = 4
Me.txtbx5.Text = ""
'
'txtbx6

Me.txtbx6.Location = New System.Drawing.Point(192, 184)
Me.txtbx6.Name = "txtbx6"
Me.txtbx6.Size = New System.Drawing.Size(96, 20)
Me.txtbx6.TabIndex = 5
Me.txtbx6.Text = ""

'txtbx7

Me.txtbx7.Location = New System.Drawing.Point(192, 208)
Me.txtbx7.Name = "txtbx7"
Me.txtbx7.Size = New System.Drawing.Size(96, 20)
Me.txtbx7.TabIndex = 6
Me.txtbx7.Text = ""

'lbl1
'
Me.lbl1.Location = New System.Drawing.Point(24, 64)
Me.lbl1.Name = "lbl1"
Me.lbl1.Size = New System.Drawing.Size(56, 23)
Me.lbl1.TabIndex = 7
Me.lbl1.Text = "Channel 1"

'lbl2

Me.lbl2.Location = New System.Drawing.Point(24, 88)
Me.lbl2.Name = "lbl2"
Me.lbl2.Size = New System.Drawing.Size(56, 23)
Me.lbl2.TabIndex = 8
Me.lbl2.Text = "Channel 2"

'lbl3
'
Me.lbl3.Location = New System.Drawing.Point(24, 112)
Me.lbl3.Name = "lbl3"
Me.lbl3.Size = New System.Drawing.Size(56, 24)
Me.lbl3.TabIndex = 9
Me.lbl3.Text = "Channel 3"
'
'lbl4

Me.lbl4.Location = New System.Drawing.Point(24, 136)
Me.lbl4.Name = "lbl4"
Me.lbl4.Size = New System.Drawing.Size(56, 23)
```

```
        Me.lbl4.TabIndex = 10
        Me.lbl4.Text = "Channel 4"
        '
        'lbl5
        '
        Me.lbl5.Location = New System.Drawing.Point(24, 160)
        Me.lbl5.Name = "lbl5"
        Me.lbl5.Size = New System.Drawing.Size(56, 23)
        Me.lbl5.TabIndex = 11
        Me.lbl5.Text = "Channel 5"
        '
        'lbl6
        '
        Me.lbl6.Location = New System.Drawing.Point(24, 184)
        Me.lbl6.Name = "lbl6"
        Me.lbl6.Size = New System.Drawing.Size(56, 23)
        Me.lbl6.TabIndex = 12
        Me.lbl6.Text = "Channel 6"
        '
        'lbl7
        '
        Me.lbl7.Location = New System.Drawing.Point(24, 208)
        Me.lbl7.Name = "lbl7"
        Me.lbl7.Size = New System.Drawing.Size(56, 23)
        Me.lbl7.TabIndex = 13
        Me.lbl7.Text = "Channel 7"
        '
        'txtbx0
        '
        Me.txtbx0.Location = New System.Drawing.Point(192, 40)
        Me.txtbx0.Name = "txtbx0"
        Me.txtbx0.Size = New System.Drawing.Size(96, 20)
        Me.txtbx0.TabIndex = 14
        Me.txtbx0.Text = ""
        '
        'lbl0
        '
        Me.lbl0.Location = New System.Drawing.Point(24, 40)
        Me.lbl0.Name = "lbl0"
        Me.lbl0.Size = New System.Drawing.Size(56, 23)
        Me.lbl0.TabIndex = 15
        Me.lbl0.Text = "Channel 0"
        '
        'btnStart
        '
        Me.btnStart.Location = New System.Drawing.Point(408, 64)
        Me.btnStart.Name = "btnStart"
        Me.btnStart.Size = New System.Drawing.Size(104, 23)
        Me.btnStart.TabIndex = 16
        Me.btnStart.Text = "Start System"
        '
        'Label9
        '
        Me.Label9.Location = New System.Drawing.Point(192, 24)
        Me.Label9.Name = "Label9"
        Me.Label9.Size = New System.Drawing.Size(88, 16)
        Me.Label9.TabIndex = 34
        Me.Label9.Text = "Seconds"
        '
        'btnReset
        '
        Me.btnReset.Location = New System.Drawing.Point(408, 112)
        Me.btnReset.Name = "btnReset"
        Me.btnReset.Size = New System.Drawing.Size(104, 23)
        Me.btnReset.TabIndex = 36
        Me.btnReset.Text = "Reset"
        '
        'Label11
        '
        Me.Label11.Font = New System.Drawing.Font("Monaco", 18.0!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
```

```
        Me.Label11.Location = New System.Drawing.Point(40, 8)
        Me.Label11.Name = "Label11"
        Me.Label11.Size = New System.Drawing.Size(504, 40)
        Me.Label11.TabIndex = 38
        Me.Label11.Text = "Gel Loading Timer Program"
        '
        'GroupBox1
        '
        Me.GroupBox1.Controls.Add(Me.btnManual)
        Me.GroupBox1.Controls.Add(Me.btnDepress)
        Me.GroupBox1.Controls.Add(Me.lblPressure)
        Me.GroupBox1.Controls.Add(Me.btnInit)
        Me.GroupBox1.Controls.Add(Me.Label2)
        Me.GroupBox1.Controls.Add(Me.txtMin7)
        Me.GroupBox1.Controls.Add(Me.txtMin6)
        Me.GroupBox1.Controls.Add(Me.txtMin5)
        Me.GroupBox1.Controls.Add(Me.txtMin4)
        Me.GroupBox1.Controls.Add(Me.txtMin3)
        Me.GroupBox1.Controls.Add(Me.txtMin2)
        Me.GroupBox1.Controls.Add(Me.txtMin1)
        Me.GroupBox1.Controls.Add(Me.txtMin0)
        Me.GroupBox1.Controls.Add(Me.Label1)
        Me.GroupBox1.Controls.Add(Me.txtVolt7)
        Me.GroupBox1.Controls.Add(Me.txtVolt6)
        Me.GroupBox1.Controls.Add(Me.txtVolt5)
        Me.GroupBox1.Controls.Add(Me.txtVolt4)
        Me.GroupBox1.Controls.Add(Me.txtVolt3)
        Me.GroupBox1.Controls.Add(Me.txtVolt2)
        Me.GroupBox1.Controls.Add(Me.txtVolt1)
        Me.GroupBox1.Controls.Add(Me.txtVolt0)
        Me.GroupBox1.Controls.Add(Me.txtbx1)
        Me.GroupBox1.Controls.Add(Me.txtbx6)
        Me.GroupBox1.Controls.Add(Me.txtbx7)
        Me.GroupBox1.Controls.Add(Me.lbl1)
        Me.GroupBox1.Controls.Add(Me.lbl2)
        Me.GroupBox1.Controls.Add(Me.lbl3)
        Me.GroupBox1.Controls.Add(Me.lbl4)
        Me.GroupBox1.Controls.Add(Me.lbl5)
        Me.GroupBox1.Controls.Add(Me.lbl6)
        Me.GroupBox1.Controls.Add(Me.lbl7)
        Me.GroupBox1.Controls.Add(Me.txtbx0)
        Me.GroupBox1.Controls.Add(Me.lbl0)
        Me.GroupBox1.Controls.Add(Me.btnStart)
        Me.GroupBox1.Controls.Add(Me.Label9)
        Me.GroupBox1.Controls.Add(Me.btnReset)
        Me.GroupBox1.Controls.Add(Me.txtbx5)
        Me.GroupBox1.Controls.Add(Me.txtbx2)
        Me.GroupBox1.Controls.Add(Me.txtbx3)
        Me.GroupBox1.Controls.Add(Me.txtbx4)
        Me.GroupBox1.Location = New System.Drawing.Point(8, 48)
        Me.GroupBox1.Name = "GroupBox1"
        Me.GroupBox1.Size = New System.Drawing.Size(536, 240)
        Me.GroupBox1.TabIndex = 39
        Me.GroupBox1.TabStop = False
        Me.GroupBox1.Text = "Gel System"
        '
        'lblPressure
        '
        Me.lblPressure.Font = New System.Drawing.Font("Albertus Extra Bold", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.lblPressure.Location = New System.Drawing.Point(416, 208)
        Me.lblPressure.Name = "lblPressure"
        Me.lblPressure.Size = New System.Drawing.Size(96, 16)
        Me.lblPressure.TabIndex = 59
        Me.lblPressure.Text = "Depressurized"
        '
        'btnInit
        '
        Me.btnInit.Location = New System.Drawing.Point(408, 40)
        Me.btnInit.Name = "btnInit"
        Me.btnInit.Size = New System.Drawing.Size(104, 23)
```

```
Me.btnInit.TabIndex = 57
Me.btnInit.Text = "Initialize System"
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(88, 24)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(88, 16)
Me.Label2.TabIndex = 56
Me.Label2.Text = "Minutes"
'
'txtMin7
'
Me.txtMin7.Location = New System.Drawing.Point(88, 208)
Me.txtMin7.Name = "txtMin7"
Me.txtMin7.Size = New System.Drawing.Size(96, 20)
Me.txtMin7.TabIndex = 55
Me.txtMin7.Text = ""
'
'txtMin6
'
Me.txtMin6.Location = New System.Drawing.Point(88, 184)
Me.txtMin6.Name = "txtMin6"
Me.txtMin6.Size = New System.Drawing.Size(96, 20)
Me.txtMin6.TabIndex = 54
Me.txtMin6.Text = ""
'
'txtMin5
'
Me.txtMin5.Location = New System.Drawing.Point(88, 160)
Me.txtMin5.Name = "txtMin5"
Me.txtMin5.Size = New System.Drawing.Size(96, 20)
Me.txtMin5.TabIndex = 53
Me.txtMin5.Text = ""

'txtMin4

Me.txtMin4.Location = New System.Drawing.Point(88, 136)
Me.txtMin4.Name = "txtMin4"
Me.txtMin4.Size = New System.Drawing.Size(96, 20)
Me.txtMin4.TabIndex = 52
Me.txtMin4.Text = ""

'txtMin3
'
Me.txtMin3.Location = New System.Drawing.Point(88, 112)
Me.txtMin3.Name = "txtMin3"
Me.txtMin3.Size = New System.Drawing.Size(96, 20)
Me.txtMin3.TabIndex = 51
Me.txtMin3.Text = ""
'
'txtMin2
'
Me.txtMin2.Location = New System.Drawing.Point(88, 88)
Me.txtMin2.Name = "txtMin2"
Me.txtMin2.Size = New System.Drawing.Size(96, 20)
Me.txtMin2.TabIndex = 50
Me.txtMin2.Text = ""
'
'txtMin1
'
Me.txtMin1.Location = New System.Drawing.Point(88, 64)
Me.txtMin1.Name = "txtMin1"
Me.txtMin1.Size = New System.Drawing.Size(96, 20)
Me.txtMin1.TabIndex = 49
Me.txtMin1.Text = ""
'
'txtMin0
'
Me.txtMin0.Location = New System.Drawing.Point(88, 40)
Me.txtMin0.Name = "txtMin0"
```

```
Me.txtMin0.Size = New System.Drawing.Size(96, 20)
Me.txtMin0.TabIndex = 48
Me.txtMin0.Text = ""
'
'Label1
'
Me.Label1.Location = New System.Drawing.Point(296, 24)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(100, 16)
Me.Label1.TabIndex = 47
Me.Label1.Text = "Channel Voltages"
'
'txtVolt7
'
Me.txtVolt7.Location = New System.Drawing.Point(296, 208)
Me.txtVolt7.Name = "txtVolt7"
Me.txtVolt7.TabIndex = 46
Me.txtVolt7.Text = ""
'
'txtVolt6
'
Me.txtVolt6.Location = New System.Drawing.Point(296, 184)
Me.txtVolt6.Name = "txtVolt6"
Me.txtVolt6.TabIndex = 45
Me.txtVolt6.Text = ""

  'txtVolt5

Me.txtVolt5.Location = New System.Drawing.Point(296, 160)
Me.txtVolt5.Name = "txtVolt5"
Me.txtVolt5.TabIndex = 44
Me.txtVolt5.Text = ""
'
'txtVolt4
'
Me.txtVolt4.Location = New System.Drawing.Point(296, 136)
Me.txtVolt4.Name = "txtVolt4"
Me.txtVolt4.TabIndex = 43
Me.txtVolt4.Text = ""
'
'txtVolt3
'
Me.txtVolt3.Location = New System.Drawing.Point(296, 112)
Me.txtVolt3.Name = "txtVolt3"
Me.txtVolt3.TabIndex = 42
Me.txtVolt3.Text = ""
'
'txtVolt2
'
Me.txtVolt2.Location = New System.Drawing.Point(296, 88)
Me.txtVolt2.Name = "txtVolt2"
Me.txtVolt2.TabIndex = 41
Me.txtVolt2.Text = ""

'txtVolt1
'
Me.txtVolt1.Location = New System.Drawing.Point(296, 64)
Me.txtVolt1.Name = "txtVolt1"
Me.txtVolt1.TabIndex = 40
Me.txtVolt1.Text = ""
'
'txtVolt0

Me.txtVolt0.Location = New System.Drawing.Point(296, 40)
Me.txtVolt0.Name = "txtVolt0"
Me.txtVolt0.TabIndex = 39
Me.txtVolt0.Text = ""
'
'btnDepress
'
Me.btnDepress.Location = New System.Drawing.Point(408, 160)
```

```
        Me.btnDepress.Name = "btnDepress"
        Me.btnDepress.Size = New System.Drawing.Size(104, 23)
        Me.btnDepress.TabIndex = 60
        Me.btnDepress.Text = "Depressurize"
        '
        'btnManual
        '
        Me.btnManual.Location = New System.Drawing.Point(408, 136)
        Me.btnManual.Name = "btnManual"
        Me.btnManual.Size = New System.Drawing.Size(104, 23)
        Me.btnManual.TabIndex = 61
        Me.btnManual.Text = "Manual Pressure"
        '
        'Form1
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(552, 294)
        Me.Controls.Add(Me.GroupBox1)
        Me.Controls.Add(Me.Label11)
        Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        Me.Name = "Form1"
        Me.Text = "Gel Loading Timer Program by Nate Ball"
        Me.GroupBox1.ResumeLayout(False)
        Me.ResumeLayout(False)

    End Sub

#End Region

        'This line creates and initializes an instance of the [illegible] wrapper class
    Dim myAgilent66xx As Agilent.Agilent66xx.Interop.Agilent66xx

     'These DLL calls activate functions from NIDAQ to do single-point read values for
     individual channels from the DAQ card
    Declare Function AI_VRead Lib "nidaq32.dll" (ByVal deviceNumber As Int16, ByVal chan
As Int16, ByVal gain As Int16, ByRef voltage As Double) As Int16
    Declare Function AO_Update Lib "nidaq32.dll" (ByVal deviceNumber As Int16) As Int16
    Declare Function AO_VWrite Lib "nidaq32.dll" (ByVal deviceNUmber As Int16, ByVal chan
As Int16, ByVal voltage As Double) As Int16

    '[illegible] the voltage variables
    Dim voltin0 As Double
    Dim voltin1 As Double
    Dim voltin2 As Double
    Dim voltin3 As Double
    Dim voltin4 As Double
    Dim voltin5 As Double
    Dim voltin6 As Double
    Dim voltin7 As Double
    Dim voltin99 As Double

     '[illegible] variable used to store single-point voltage before [illegible]
     voltage variable
    Dim status As Int16

    Public dateStartm As Integer
    Public dateStartms As Integer
    Public dateStarts As Integer

    '[illegible] components that are [illegible] for each channel to create stopwatch
    Private date0m As Integer
    Private date0ms As Integer
    Private date0s As Integer
    Private date1m As Integer
    Private date1ms As Integer
    Private date1s As Integer
    Private date2m As Integer
    Private date2ms As Integer
    Private date2s As Integer
    Private date3m As Integer
    Private date3ms As Integer
```

```
    Private date3s As Integer
    Private date4m As Integer
    Private date4ms As Integer
    Private date4s As Integer
    Private date5m As Integer
    Private date5ms As Integer
    Private date5s As Integer
    Private date6m As Integer
    Private date6ms As Integer
    Private date6s As Integer
    Private date7m As Integer
    Private date7ms As Integer
    Private date7s As Integer

    'status of channels. True = filled, False = unfilled
    Dim h As Boolean = False
    Dim i As Boolean = False
    Dim j As Boolean = False
    Dim k As Boolean = False
    Dim l As Boolean = False
    Dim m As Boolean = False
    Dim n As Boolean = False
    Dim o As Boolean = False


    Private Sub btnStart_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnStart.Click

        'This file turns activates the power supply that runs the electric valve
        'pressurizing the system
        myAgilent66xx.Output.VoltageLevel = 24
        myAgilent66xx.Output.CurrentLimit = 0.2
        myAgilent66xx.Output.Enabled = True

        'start times that will be subtracted from each channel's individual datetime
        'to get time to fill
        dateStartm = Date.Now.Minute
        dateStarts = Date.Now.Second
        dateStartms = Date.Now.Millisecond

        lblPressure.Text = "PRESSURIZED"
        lblPressure.Refresh()

        'This while statement keeps scanning the channels until they are all filled.
        'If the channel fills, it changes to True, and the program skips it in the next
        'loop to see if all channels are full. The system stops.
        While Not ((h And i And j And k And l And m And n And o) = True)

            If h = False Then
                status = AI_VRead(2, 0, 1, voltin0)
                If voltin0 > 0.1 Then 'threshold voltage that indicates full channel
                    date0m = Date.Now.Minute
                    date0s = Date.Now.Second 'records immediate datetime if channel is
full
                    date0ms = Date.Now.Millisecond
                    txtVolt0.Text = voltin0 'displays voltage in textbox
                    txtMin0.Text = (date0m - dateStartm)
                    'subtracts start datetime from end datetime in order to get full
time
                    txtbx0.Text = ((date0s - dateStarts) + 0.001 * (date0ms -
dateStartms))
                    h = True 'channel is now full
                    txtVolt0.Refresh()
                    txtMin0.Refresh()
                    txtbx0.Refresh()
                End If
            End If

            If i = False Then
                status = AI_VRead(2, 1, 1, voltin1)
                If voltin1 > 0.1 Then
```

```
                    date1m = Date.Now.Minute
                    date1s = Date.Now.Second
                    date1ms = Date.Now.Millisecond
                    txtVolt1.Text = voltin1
                    txtMin1.Text = (date1m - dateStartm)
                    txtbx1.Text = ((date1s - dateStarts) + 0.001 * (date1ms -
dateStartms))
                    i = True
                    txtVolt1.Refresh()
                    txtMin1.Refresh()
                    txtbx1.Refresh()
                End If
            End If

            If j = False Then
                status = AI_VRead(2, 2, 1, voltin2)
                If voltin2 > 0.1 Then
                    date2m = Date.Now.Minute
                    date2s = Date.Now.Second
                    date2ms = Date.Now.Millisecond
                    txtVolt2.Text = voltin2
                    txtMin2.Text = (date2m - dateStartm)
                    txtbx2.Text = ((date2s - dateStarts) + 0.001 * (date2ms -
dateStartms))
                    j = True
                    txtVolt2.Refresh()
                    txtMin2.Refresh()
                    txtbx2.Refresh()
                End If
            End If

            If k = False Then
                status = AI_VRead(2, 3, 1, voltin3)
                If voltin3 > 0.1 Then
                    date3m = Date.Now.Minute
                    date3s = Date.Now.Second
                    date3ms = Date.Now.Millisecond
                    txtVolt3.Text = voltin3
                    txtMin3.Text = (date3m - dateStartm)
                    txtbx3.Text = ((date3s - dateStarts) + 0.001 * (date3ms -
dateStartms))
                    k = True
                    txtVolt3.Refresh()
                    txtMin3.Refresh()
                    txtbx3.Refresh()
                End If
            End If

            If l = False Then
                status = AI_VRead(2, 4, 1, voltin4)
                If voltin4 > 0.1 Then
                    date4m = Date.Now.Minute
                    date4s = Date.Now.Second
                    date4ms = Date.Now.Millisecond
                    txtVolt4.Text = voltin4
                    txtMin4.Text = (date4m - dateStartm)
                    txtbx4.Text = ((date4s - dateStarts) + 0.001 * (date4ms -
dateStartms))
                    l = True
                    txtVolt4.Refresh()
                    txtMin4.Refresh()
                    txtbx4.Refresh()
                End If
            End If

            If m = False Then
                status = AI_VRead(2, 5, 1, voltin5)
                If voltin5 > 0.1 Then
                    date5m = Date.Now.Minute
                    date5s = Date.Now.Second
                    date5ms = Date.Now.Millisecond
```

```
                    txtVolt5.Text = voltin5
                    txtMin5.Text = (date5m - dateStartm)
                    txtbx5.Text = ((date5s - dateStarts) + 0.001 * (date5ms -
dateStartms))
                    m = True
                    txtVolt5.Refresh()
                    txtMin5.Refresh()
                    txtbx5.Refresh()
                End If
            End If

            If n = False Then
                status = AI_VRead(2, 6, 1, voltin6)
                If voltin6 > 0.1 Then
                    date6m = Date.Now.Minute
                    date6s = Date.Now.Second
                    date6ms = Date.Now.Millisecond
                    txtVolt6.Text = voltin6
                    txtMin6.Text = (date6m - dateStartm)
                    txtbx6.Text = ((date6s - dateStarts) + 0.001 * (date6ms -
dateStartms))
                    n = True
                    txtVolt6.Refresh()
                    txtMin6.Refresh()
                    txtbx6.Refresh()
                End If
            End If

            If o = False Then
                status = AI_VRead(2, 7, 1, voltin7)
                If voltin7 > 0.1 Then
                    date7m = Date.Now.Minute
                    date7s = Date.Now.Second
                    date7ms = Date.Now.Millisecond
                    txtVolt7.Text = voltin7
                    txtMin7.Text = (date7m - dateStartm)
                    txtbx7.Text = ((date7s - dateStarts) + 0.001 * (date7ms -
dateStartms))
                    o = True
                    txtVolt7.Refresh()
                    txtMin7.Refresh()
                    txtbx7.Refresh()
                End If
            End If

        End While

    End Sub


    Private Sub btnReset_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnReset.Click
        ' Keep power supply, turning off pressure to system
        myAgilent66xx.Output.VoltageLevel = 0
        myAgilent66xx.Output.CurrentLimit = 0
        ' Set all textboxes to 0
        txtbx0.Text = 0.0
        txtbx1.Text = 0.0
        txtbx2.Text = 0.0
        txtbx3.Text = 0.0
        txtbx4.Text = 0.0
        txtbx5.Text = 0.0
        txtbx6.Text = 0.0
        txtbx7.Text = 0.0

        txtMin0.Text = 0
        txtMin1.Text = 0
        txtMin2.Text = 0
        txtMin3.Text = 0
        txtMin4.Text = 0
        txtMin5.Text = 0
```

```
        txtMin6.Text = 0
        txtMin7.Text = 0

        txtVolt0.Text = 0
        txtVolt1.Text = 0
        txtVolt2.Text = 0
        txtVolt3.Text = 0
        txtVolt4.Text = 0
        txtVolt5.Text = 0
        txtVolt6.Text = 0
        txtVolt7.Text = 0

        lbl0.Text = "Channel 0"
        lbl1.Text = "Channel 1"
        lbl2.Text = "Channel 2"
        lbl3.Text = "Channel 3"
        lbl4.Text = "Channel 4"
        lbl5.Text = "Channel 5"
        lbl6.Text = "Channel 6"
        lbl7.Text = "Channel 7"
        [illegible]
        h = False
        i = False
        j = False
        k = False
        l = False
        m = False
        n = False
        o = False

        lblPressure.Text = "Depressurized"
        lblPressure.Refresh()

    End Sub


    Private Sub btnInit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnInit.Click
        [illegible]
        myAgilent66xx = New Agilent.Agilent66xx.Interop.Agilent66xxClass
        myAgilent66xx.Initialize("GPIB0::1::INSTR", True, True, Nothing)
    End Sub


    Private Sub btnDepress_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDepress.Click
        [illegible]
[illegible]
        myAgilent66xx.Output.VoltageLevel = 0
        myAgilent66xx.Output.CurrentLimit = 0
        lblPressure.Text = "Depressurized"
        lblPressure.Refresh()
    End Sub

    Private Sub btnManual_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnManual.Click
        [illegible]
        myAgilent66xx.Output.VoltageLevel = 24
        myAgilent66xx.Output.CurrentLimit = 0.2
        myAgilent66xx.Output.Enabled = True
        lblPressure.Text = "PRESSURIZED"
        lblPressure.Refresh()
    End Sub
End Class
```