DESIGN AND FABRICATION OF A PRECISE AND AUTOMATED SYSTEM FOR THE DEPOSITION OF MICROLITER FLUID VOLUMES.

A Thesis
Presented to the Faculty of the Graduate School
Of Cornell University
In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering

by
William Pottle
August 2002

ABSTRACT

Membrane strip-type optical biosensors consist of a nucleic acid capture probe immobilized onto a nitrocellulose membrane. The target analyte is then mixed with marker-encapsulated liposomes tagged with a sequence complimentary to one section of the analyte. The mixture migrates through the capture zone, where only analytes that have a sequence complimentary to that of the capture probe remain bound. The analyte concentration is then proportional to the marker signal recorded. The development of these biosensors requires spotting 1 µL of fluid on each 5 mm strip. Each strip then undergoes a tedious immobilization and blocking procedure. An automated system similar to that of an inkjet printer was developed that deposits fluid on one 18.5 cm sheet of membrane at a time in one continuous line- the one sheet is taken through the immobilization and blocking process, and subsequently cut into individual strips. The fluid system consists of a stepper motor mechanically connected to a syringe. This piece is mounted on the print head of an inkjet printer. Both motors are controlled by 293D driver chips and control input is provided directly by a National Instruments 6503 Digital I/O board. The board settings and timing are controlled by a C++ program. The system is more rapid than current procedures and reduces the variability in sensors produced by different rsearchers. When run under the default parameters, the system gives an average deposition of 37.1 µL per sweep, with a standard deviation of 2.83 µL. The system lays the fluid down in a line with an average width of 1.91 mm and a standard deviation of 0.54 mm. Membranes made with the system compare well against membranes made by hand. Their background noise level is comparable, while on average the membranes made with the automated system give a 35% higher signal and

have a 7% lower standard deviation. The higher signal is likely due to the fluid line having a higher concentration of capture probes and the technician being able to fit a larger portion of the signal into the reflectometer opening. The user can make approximately 320 membrane strips before refilling the device. The proposed system can provide a relatively inexpensive method for the deposition of lines of fluid in the laboratory setting.

BIOGRAPHICAL SKETCH

William Pottle was grew up in Denver, CO where he was born on July 12, 1978. He graduated from Overland High School in 1997. He received his B.S. degree from Cornell in Agricultural and Biological Engineering in 2001. While at Cornell he helped the Cornell Taekwondo team earn five more Ivy League championships. His hobbies include martial arts, writing, breakdancing and soccer.

ACKNOWLEDGMENTS

The author acknowledges the significant help of several individuals. First and foremost, the author would like to thank his parents for many years of support and always putting education first when he was growing up. Professor Antje Baeumner gave overall guidance, direction, and financial support. Doug Caveney in the Riley Robb machine shop was instrumental in helping to design and construct the fluid deposition system and the motor/syringe interface. Sean Breheny was also helpful in designing the control circuit and suggested the use of the H bridge. John Reilly and Eric Hoffman helped by making membranes for comparison with the established protocol. Jeff Lee and Sherry Lai also provided assistance at various stages of the project.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

V. Introduction

V.1 Biosensors

Biosensors are important devices that provide information about a specific analyte present in a particular sample. Advantages of biosensors can include their ease of use, portability, reliability, sensitivity, selectivity, and low cost. They generally consist of a specific biorecognition element that recognizes the analyte by binding to it and holding on to it (affinity biorecognition elements) or by catalyzing a reaction (catalysis biorecognition elements.) The biorecognition element is responsible for biosensors' high levels of specificity. Examples of biorecognition elements include enzymes, nucleic acids, antibodies, cell surface receptors, whole cells, and carbohydrates. Next, a transducer transforms the signal into a readout signal. Usually, this is in the form of a digital signal that can be quantitatively related to the original amount of analyte present. There is a wide diversity of transducers in use, including electrochemical systems, quartz piezocrystals, and optical detection mechanisms such as colorimetric, fluorimetric, refractive, etc. detection. Other factors to consider that are integral to the design and construction of biosensors include any pre-transduction signal amplification steps that need to be completed and the immobilization of the biorecognition element.

Although not all biosensors use immobilized biorecognition elements, in the majority of sensors the biorecognition element is immobilized on either the transducer surface (to lessen diffusion time) or in close proximity to it. Immobilization of a biorecognition element can also serve as a means of separating the analyte from the rest of the sample. There are several methods of immobilization including direct adsorption, covalent binding of residues in the biorecognition element to groups on the immobilization surface, and the use of linking molecules. One such case is the use of biotin and streptavidin to immobilize nucleic acid probes.

Biotin is a cofactor required for many enzymes[1] and some bacteria including *Streptomyces avidinii* have evolved the streptavidin molecule as a defense mechanism against cytolytic enzymes.[2,3] Streptavidin is a 60 kDa protein that can be covalently attached to surfaces. Biotin can be easily coupled to nucleic acids at the 5' end. Biotin and streptavidin have a high affinity for each other. The Ka value is approximately 10^15 M^{-1}, the highest of any two molecules found in nature.[2,4] One Streptavidin molecule can bind up to four biotin molecules at once.

V.2 Optical DNA Biosensors

The Bioanalytical Microsystems and Biosensors lab at Cornell University currently develops many optical DNA biosensors for the detection of pathogenic organisms such as *Cryptosporidium parvum*, *E.coli*, HIV, and four serotypes of the dengue virus.

The biosensors are made from strips of polyethersulfone membrane that are 5 mm wide and approximately 7.5 cm long. The strips contain a capture zone in which biotinylated capture probes are immobilized on the membrane. A schematic of a typical strip is shown in figure 1. A second probe, the reporter probe, is immobilized to the outside of dye-encapsulating liposomes. Depending on the analyte and its matrix, various sample preparation steps are carried out. These may include cell lysis, filtration, RNA extraction, and Nucleic Acid Sequence Based Amplification (NASBA).[5] NASBA is an isothermal molecular biological amplification technique for RNA molecules based on transcription.

Subsequent to the sample preparation steps, the target analyte (for example pathogenic RNA amplified with NASBA) is mixed with the liposomes in a hybridization buffer and applied to the membrane. The mixture moves along the membrane by capillary action with the help of a running buffer and if the target RNA sequences are complimentary to the capture and reporter probes then a sandwich is formed of reporter probe, RNA and capture probe in the capture (or detection) zone. The basic idea of this sandwich is depicted in figure 2. Non-complimentary sequences are washed away since they continue the migration along the membrane strip and are not caught in the detection zone. The intensity of the dye in the detection zone is quantified using a reflectometer and within the dynamic range of the biosensor the signal is directly proportional to the original concentration of the pathogen.

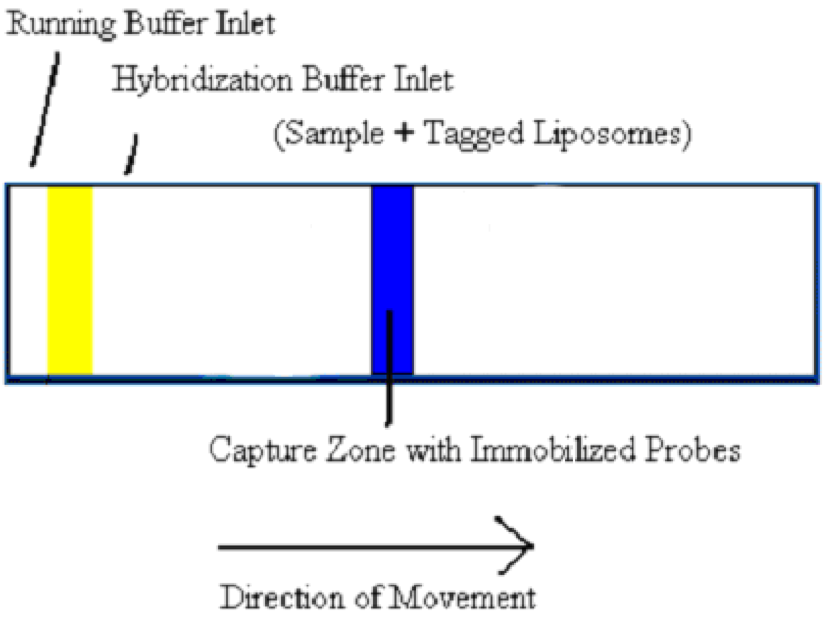

Running Buffer Inlet
Hybridization Buffer Inlet
(Sample + Tagged Liposomes)
Capture Zone with Immobilized Probes
Direction of Movement

Figure 1- Schematic of the optical DNA biosensor developed by the Bioanalytical Microsystems and Biosensors lab at Cornell. The target analyte is first mixed with the tagged liposomes in the hybridization buffer, and then added to the membrane. The rectangle between the hybridization and running buffer inlets represents a piece of tape or adhesive membrane backing that serves to force the running buffer into the membrane and make sure the buffer does not sit on top of the membrane.

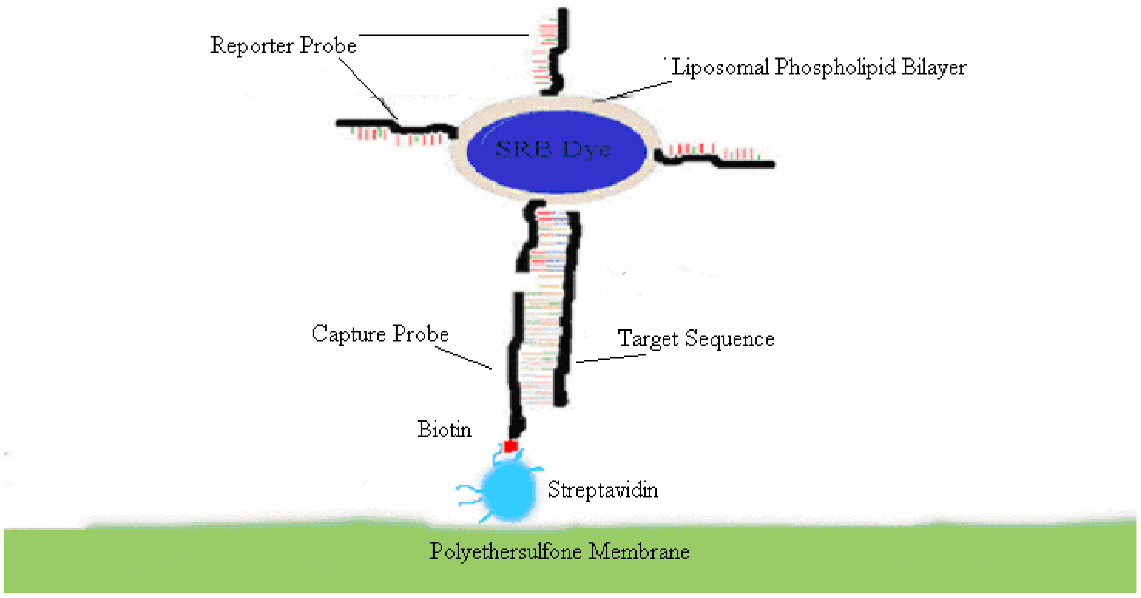


Figure 2- Not to scale schematic of the hybridization that occurs in the optical biosensor. The streptavidin molecule is bound to the polyethersulfone membrane. The biotinylated capture probe is bound to the streptavidin through the biotin/streptavidin interaction. If the target sequence has a portion complimentary to the reporter probe, it will be bound to the dye encapsulating liposome. If the target sequence is also complimentary to the capture probe, the analyte/liposome complex will be ‘captured’ and the dye intensity at a particular capture zone can be read with a reflectometer.

A key advantage of these types of biosensors is their specificity. Several levels of specificity are integrated in the biosensor to avoid any false positive signals. Searching for RNA ensures that in the case of non-viral targets, only active organisms will be detected. The intelligent design of the several NASBA primers coupled with the choice of a target gene region that has little or no cross-species homology further incases specificity. Finally, assay conditions such as optimizing the stringency of the hybridization and running buffers and blocking the membrane all ensure the reduction of non-specific binding.

V.3 Fabrication Protocol

A large sheet of polyethersulfone membrane is cut into many strips 5 mm wide. To establish the dynamic range of a biosensor, and to optimize running buffer conditions, several hundred strips can be required. Each strip must be carefully handled to avoid damaging or contaminating the membrane. The membranes are all taped to a piece of cardboard that has markings made for the locations of the capture zones. The capture zones must be located with a high degree of reproducibility as the signal is read at a certain position. The technician must pipette 1 μL of fluid and drop it in the middle of each strip, careful to obtain even coverage. If one end of the strip does not receive even coverage, then liposomes that run by that side of the strip will not be captured, even if they contain the correct analyte sequence. Then, each strip is transferred to dry under the hood for five minutes and into the vacuum oven to dry for one and a half hours. Next, the strips must be soaked in a blocking reagent to block unspecific binding of the liposomes to the membrane surface. The membranes must be gently dropped onto the surface and sink only after water has been drawn into the sides by capillary action, otherwise there is a chance that the solution will not cover the membrane equally. After 30 min on the shaker at room temperature, each strip has to be removed with tweezers and blotted dry with Kimwipes. Finally, they are again attached to the cardboard with tape and allowed to dry for five minutes under the hood and for three to four hours in the vacuum oven.

V.4 Project Motivation

The system described in this paper has several key advantages over the current protocol. The automated system is more accurate because it dispenses fluid evenly with a uniform rate and is not dependant on the skill of the technician. The described system also makes great strides in the ease of the entire project by depositing the fluid on one large 20.5 cm by 7.5 cm sheet instead of many small strips. Now, instead of handling each strip with tweezers and waiting for it to sink in the blocking solution, the entire process can be handled by moving only one sheet at a time. Not only does this reduce contamination, but it also can reduce the time required to make membranes dramatically. The one sheet is taken through the entire process, and then cut into strips only at the end after the final drying.

V.5 Other Microfluid Deposition Systems

There are other systems currently available, most notably from CAMAG (Berlin, Germany). It is a planar chromatography company, which offers two automated systems, the Linomat 5 and the TLC Sampler 4.[6,7] Both systems use the same technology, but the TLC Sampler 4 is a completely automated system where the Linomat 5 requires the user to manually change the target fluid and clean the syringe. The systems are similar to the system developed here. The Linomat 5 uses a syringe custom made by Hamilton Corporation (Reno, NV) and two stepper motors.[6] One motor controls the lateral position and has a step size of 0.0125 mm.[6] The motor that pushes the fluid out makes about twice as many steps per rotation. The 'print head' portion moves at approximately 10 cm/s and the system can deposit fluid on up to a 19 cm strip.[6] The system also requires nitrogen gas to operate and sprays the fluid out onto a surface in bands. The entire system is controlled through a winCATS software program that offers complete control over the system and specification of the amount of fluid to be deposited as well as the number and size of any bands. Systems of this type are capable of depositing down to approximately 100 pL of fluid.[6]

Higher resolution can be achieved through heating the liquid; some inkjet printers are capable of drop sizes on the order of 4 pL. [8] However, heating the liquid almost always results in the denaturation of proteins and nucleic acids.

In addition, several inkjet printing systems have been described in literature.[9-14, 21] These systems mostly use the existing printer setup and merely replace the ink with the desired solution to be patterned. These systems all use a commercially available inkjet printer as a printer. An image file with the desired deposition pattern is created on the computer in any graphics editing program and then the user simply selects the print command. Recent work includes a system for depositing poly(acrylic acid) to form polyelectrolyte multilayers for electroless plating of nickel.[21] A 2001 review of materials printed lists sol-gel glass materials, organic-inorganic hybrid lenses, conducting polymers, structural poylmers, and nucleic acids and proteins. These compounds have been deposited various substrates including paper, membranes, ceramics, metal, and glass.[12] Microfab, INC (Plano, TX) has used inkjet deposition of polymers and metal alloys to create optical elements, sensors, and electrical interconnects.[11] Inkjet printing has even been used before in the construction of biosensors.[10,14]

V.6 Scope of the work

The following thesis details the design, construction, and testing of an automated system for the deposition of microliter volumes of protein and nucleic acid solution onto an absorbing material such as a polyethersulfone membrane. The system automates a previously tedious step in the immobilization of biotinylated nucleic acid probes bound to streptavidin. While developed for this specific application, the system can in general deposit a volume of up 3 μL to 300 μL onto any paper-like membrane. This range can be extended several orders of magnitude by exchanging the syringe and motor components. However, for smaller volumes, the system will only be able to deposit in dots and not lines. The system was developed using the mechanical components of an Epson Stylus Color 860 inkjet printer for positioning the print head. The print head motor was disconnected electrically from the printer circuitry and instead connected to a control circuit using a 293D H bridge control chip from STM. A

700 series Hamilton Microliter syringe dispenses the liquid as a Heydon Switch and Instruments stepper motor pushes down the plunger. The syringe and stepper motor are held together and aligned by a machined plastic casing which also connects to the print head. Digital input signals come from a National Instruments Lab PC board installed in a dedicated computer system. The board control is provided by a C++ control program with a graphical user interface that lets the user select the operating parameters.

VI. System Design

VI.1 Overall design:

The fluid deposition system consisted of a microsyringe mechanically connected to a linear stepper motor. This system was held together in a machined plastic casing. This piece was set into an inkjet printer with most of its components removed. The printer's connections to its motors were severed, and the print head motor and syringe stepper motor were now both controlled by a computer through the use of a control circuit. As the print head moved back and forth, the syringe plunger pushed down and sprayed liquid onto any paper-like material inserted into the paper feed. The user could then adjust several input variables to the control software to run the system. The parameters are listed in table 1.

Table 1. The main input parameters that control the system and their meaning and use. These are the values that the user sets in the MembranePrinter.exe control program.

Variable	Meaning	Use
NumLoops	Number of times the printer moves from right to left and back	Set the desired number of loops. Fluid deposition should usually be accomplished in one loop
HeadDelay	Delay (ms) between the time the stepper motor finishes pushing down until the print head starts moving right.	This delay time will be the time it takes the print head to get to the far left side.
delayOfset	Extra delay (ms) to compensate for the time the stepper motor takes to push down	This time is necessary to ensure that the print head returns to the correct position on the right.
numSteps	Number of steps of stepper motor	Use with step delay to control the amount of fluid deposited
stepDelay	Delay between steps	Use to control the stepping rate of the motor. Thrust is a nonlinear function of stepping rate.

Figure 3 shows a photograph of the main components of the overall system, excluding the computer and control circuitry.

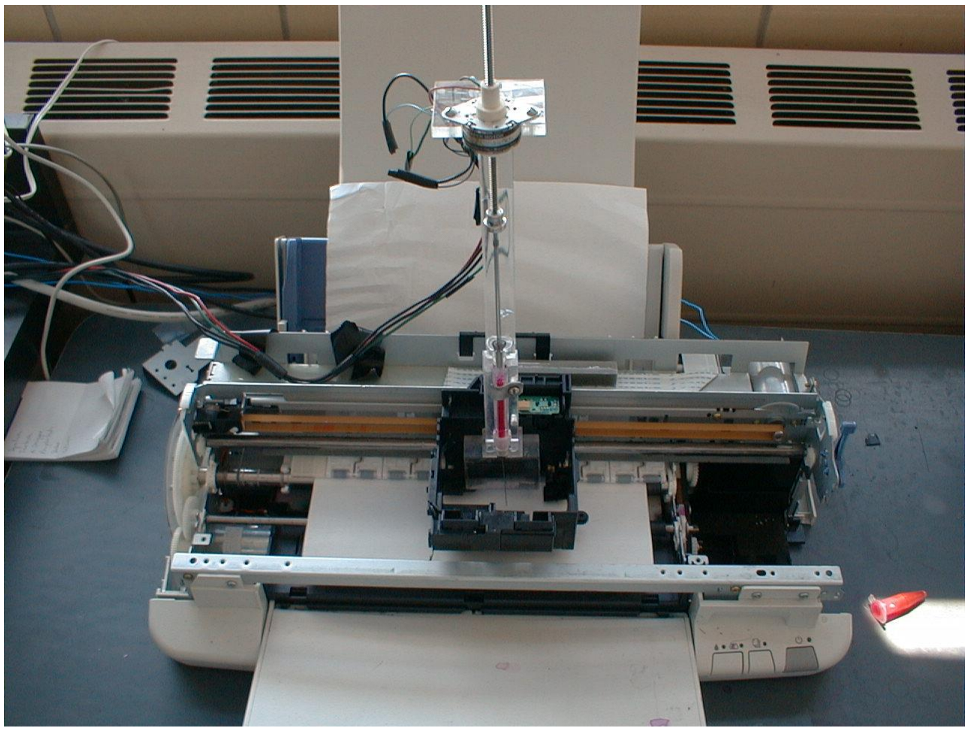


Figure 3. Photograph of overall system.

VI.2 Design Criteria

VI.2.1 Key Parameters

The project started out with a specific task in mind and the implementation of that task not defined. Several design parameters were specified based on current protocols and the equipment used, and are discussed in more detail below. Some parameters were fixed while others were flexible, and the parameters could be controlled with varying degrees of precision. For instance, the amount of fluid to be deposited and the maximum length of a run were set, while the speed of the print head or the number of loops were both flexible. The most important parameter was the amount of fluid deposited on the membrane. The amount of fluid deposited equals the volumetric flow rate divided by the linear speed of the print head:

Fluid Deposited [μL /cm] = Volumetric Flow Rate [μL /s] / Print Head Speed [cm/s]

The volumetric flow rate was dependent on the syringe volume selected as well as the speed of the linear actuator. The speed of the linear actuator was dependent in a non-linear fashion on the number of steps and the time delay between steps, as well as the resistance offered by the plunger. The print head speed depended on the applied voltage.

Length of a Run: The length that the print head moved was set at a maximum of 30 cm. In general, a longer run length is preferred, as fewer runs will be required to produce a given number of biosensors.

Volume of Fluid Deposited: Current protocols prescribe that a 1 μL drop of fluid be deposited on a 5 mm strip. This is equivalent to 40 μL of fluid being uniformly deposited on the 20 cm wide membrane.

Print Head Speed: The print head speed is controlled by the voltage applied. A voltage divider circuit inserted between the H-bridge chip and the supply rails can approximately control the applied voltage. Another way to control the voltage is to use a variable voltage power supply. This gives the user more control, but has the disadvantage of making the system more difficult to use.

Syringe Volume: The syringe volume determines how many runs can be completed before the syringe needs to be refilled. All Hamilton syringes investigated have identical lengths but varying inside diameters. Thus, the larger the syringe the more fluid is expelled for a given plunger movement. For example, on the 500 µL syringe, a 1 cm plunger depression expels 83.3 µL of fluid. On the 250 µL syringe the fluid expelled is 41.6 µL, and on the 50 µL syringe the fluid expelled is only 8.3 µL. The choice of a syringe volume is thus a balance between precision and ease of use.

VI.2.2 Figures of Merit

Ease of use: The software program should be designed with a Graphical User Interface (GUI) so that users can easily adjust parameters. Although the software must be run on a computer with a connected LABPC board, the software should be modular and should be portable to different machines for testing or for transfer of the board. The program should be a self-packaged executable not requiring recompiling. A knowledgeable user should be able to operate the device with minimal training and extensive calibration and readjustment of syringe height and operating parameters should not be required. The device should complete as many runs as possible between refills.

Non-interference with biological processes: The system must not interfere with the biological processes at work. None of the operating parameters or applied chemicals can cause denaturation of proteins or nucleic acid and the system must be as free of contamination as possible, since nuclease enzymes could digest the nucleic acid in the sample.

Repeatability: The system should always deposit the same amount of fluid onto the membrane. Thus it is especially important to make sure that the fluid deposited is the same from different parts of the syringe. The fluid ejected should be a well-defined function of the number of steps that the control program sends to the motor.

Range: The system is centered on depositing a particular volume. However, in the future needs may change. Ideally the system should be able to extend its reliable operating range to both larger and smaller volumes.

VI.2.3 Budgetary and other constraints

Due to the high cost of commercially available systems, budgetary constraints were not much of an issue. The entire project was completed on a budget of 880$, where commercial systems can cost many thousands of dollars. The CAMAG Linomat 5 retails for $6,500 and the ATS4 costs $19,500.[15] After removing costs for components that were tested but not used in the final device, the prototype cost drops to 432$.

Table 2. Budgetary Information- System components and their prices. The items are divided by which ones were actually used in the final device, and which were only used in testing.

Used Items		**Items Tested but not Used**	
Item	**Cost**	**Item**	**Cost**
Epson Stylus Color 860 Printer	$ 100.00	750SNR 500 µL Syringe (22/2"/3)	$ 42.00
PCI 6503 National Instruments Card	$ 125.00	725LT 250 µL Syringe Bottom of Form	$ 40.00
CD-50 LP Connector Block	$ 70.00	702LT 25 µL Syringe Bottom of Form	$ 24.00
725LT 250 µL Syringe	$ 40.00	1725TLL 250 µL Syringe	$ 36.00
N725 Metal Hub Needle (25/2"/3) 12/pk	$ 48.00	N733 Metal Hub Needle (33/2"/3) 12/pk	$ 80.00
Global Specialties Pro-S Lab Station	$ 64.00	293 D Chips	$ 20.00
293D Chips	$ 6.00	HIS- 26000 and 46000 Series Motors	$ 190.00
HIS 26844-12 Motor	$ 95.00		
Power Supply			
Subtotal	**$ 448.00**	**Subtotal**	**$ 432.00**
	Total Cost	$ **880.00**	

VI.3 Mechanical Fluid Deposition System

The fluid deposition system is the device that actually puts the fluid onto the membrane. Several design considerations were taken into account when deciding how the system should work. The system should be accurate and repeatable, and should require user intervention as infrequently as possible. When users are required to change or refill the syringe, it should be easy for them to do so and the performance of the system should not be strongly tied to the operator performing some complex task. The system is designed for operation only in the lab environment, but it should be stable under a wide variety of normal operating conditions. It should also avoid any harmful interference from stray electromagnetic, infrared, acoustic or other signals in the environment.

Several systems were considered to meet the challenges presented. A previous attempt by Peterson [16] was considered in detail. This approach involved using an Epson Stylus Color 860 printer and refilling the ink cartridges with the desired protein and nucleic acid solution. This attempt was motivated by the ease of simply replacing the fluid in the print cartridge and by the fact that the printer chosen was capable of drop sizes of only 4 pL.[8] This approach was complicated by the difficulty of getting such solutions inside the cartridges and by the fact that the details of the printer system are proprietary information. There was also a fear that vaporizing the droplet to shoot it onto the paper would denature the streptavidin and thus compromise the sample. The next step was to try to let the fluid flow onto the membrane through a tube through gravity flow and control the print head movement through escape commands or by printing an image of a line of varying widths. This proved difficult as the print head motion was not completely controlled by the user, and the fluid deposited was sporadic and unrepeatable. It was not possible to accurately measure how much fluid was actually deposited. Several working printer systems have been described, however, including one that prints out whole cells onto the membrane.[11] There has even been work reported that deposits nucleic acids onto a polyethersulfone membrane, similar to the task undertaken for this project. That project used an Apple StyleWriter ^{TM} II printer where the ink had been removed from the cartridge and replaced with the oligonucleotides in a phosphate buffer.[13] The decision to go with the current system was made in order to gain more control over the system and because of the challenges inherent in designing and constructing a novel device.

It was decided early on not to use the printer as a printer but merely to use its mechanical components. The connection between the computer and printer was severed and the print head motor was placed under the direct control of the user through the National Instruments DIO board. At this point, several possibilities were entertained for the fluid deposition system. One system employed a reservoir and magnetic plunger over which an electromagnet was placed. When current flows through the electromagnet, fluid would be ejected. Some thought was also given to using laboratory micropipetters with a motor that would depress the top button and deposit the fluid. These would be ideal because of their accuracy but this idea was rejected because of their cost and the difficulty in replacing the tip after each use. Only one run could be completed at a time- doing more would decrease accuracy. There are also commercially available fluid deposition systems containing stepper motors and syringes on other principles, such as piezoelectric deposition. These can be controlled through special electronic drivers.[11,17] Although these could be a great help, concern about cost and fluid deposition rate precluded their use. The final system was determined to be a motor that pushed down a syringe, all housed together and lined up by some plastic or metal components.

By taking a modular approach, components could be selected that were optimal for each particular task such as moving the syringe plunger or holding the fluid. A prototype was constructed with the optimal components that acted as a proof of concept, showing that the system could actually work at values approximate to those required. Through a series of successive experiments the supply voltages for both the stepper motor and print head motor and the five key operating parameters were sequentially optimized and components were changed if necessary to yield the final working device. The individual components are described in more detail below.

Figures 4-8 show computerized images of the setup without the stepper motor and with just a cylinder representing the syringe body. The images were made in the Computer Aided Design (CAD) program Microstation SE.

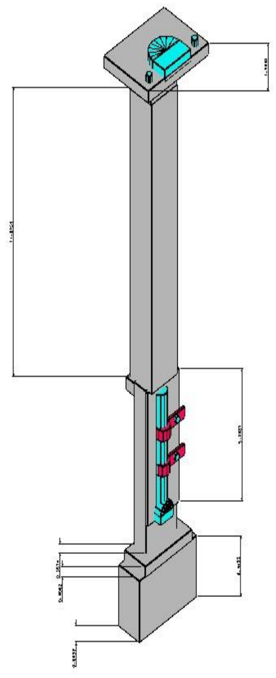


Figure 4- Isometric view of fluid deposition system. The image was rendered in Filled Hidden Line mode. The system is made from plastic except for the two dark horizontal bands which are aluminum and serve to hold the syringe in place. All dimensions are in centimeters

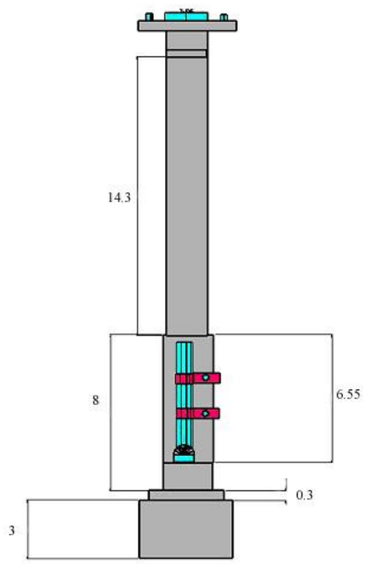


Figure 5- Front view of fluid deposition system. The image was rendered in Filled Hidden Line mode. The system is made from plastic except for the two dark horizontal bands that are aluminum and serve to hold the syringe in place. All dimensions are in centimeters.

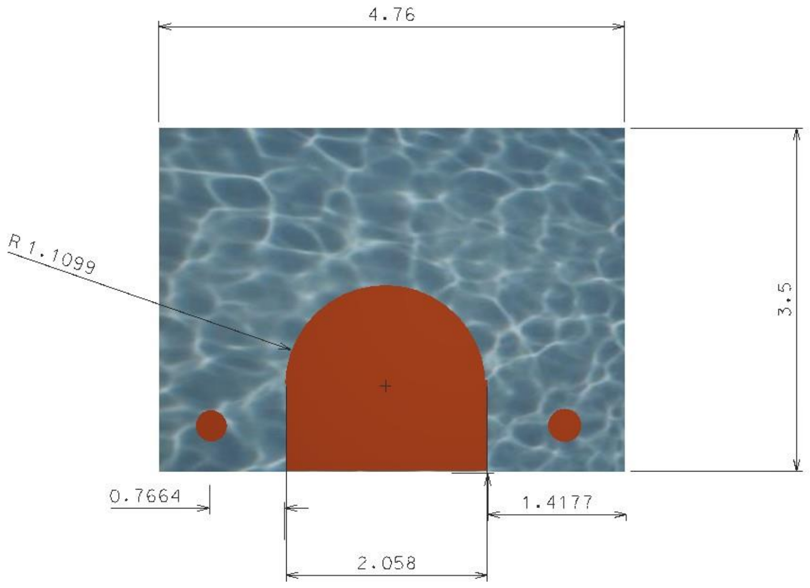


Figure 6- Top View of the motor housing unit of the device with dimensions. The main motor piece fits in the large middle hole while the two flanking holes are used for screws through the motor flanges. The image was made using the Phong shading technique. All dimensions are in centimeters.

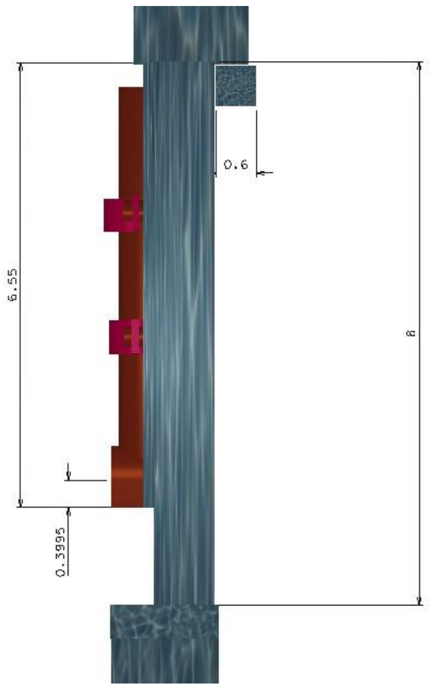


Figure 7- Side view of the lower half of the device. The syringe fits where the dark cylinder is, and the semicircle stops it from being drawn up to high into the device. The two parallel bands represent the aluminum pieces that hold the syringe in place. The image was made using the Phong shading technique. All dimensions are in centimeters.

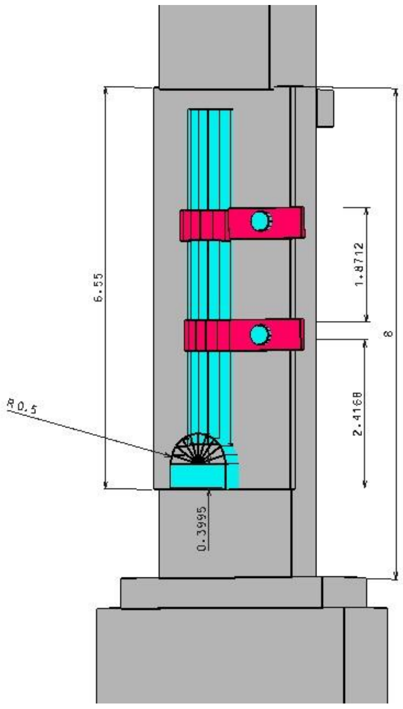


Figure 8- Rotated view of the bottom portion of the device. The syringe fits where the dark cylinder is, and the semicircle stops it from being drawn up to high into the device. The two parallel bands represent the aluminum pieces that hold the syringe in place. This image is useful to observe the different small pieces of plastic that were used. The image was made using the Filled Hidden Line shading technique. All dimensions are in centimeters.

VI.3.1 Stepper Motor

One key component of the system is the motor. The motor's main function is to push the syringe barrel down to eject the fluid. Several ideas for motor type were taken into consideration. A linear actuator was chosen over a motor with external gears because of the quality and availability of commercial models with operating ranges encompassing the desired values for thrust (>5 N), speed (~1 cm/s), and operating voltage (~12 V). Although the required software and control circuitry is more complicated for a stepper motor than a regular motor, the stepper motor was chosen for the fact that it can be precisely controlled with software. By controlling the time delay between steps, there is theoretically no lower limit on the motor speed. However, when the step delay increases beyond the critical value of about 4 ms the motion becomes choppy, and this is not desirable for depositing a thin layer of fluid. This provided an important constraint on the default operating parameters.

The motor chosen was model 26844-12 from Heydon Switch and Instrument (Waterbury, CT) The motor selected runs in bipolar mode, 12V for maximum thrust. The shaft is non-captive to allow flexibility in shaft length. The default (5 cm) shaft proved to be too short so a longer (15 cm) one was ordered. Figure 9 shows an image of the motor used with dimensions added.

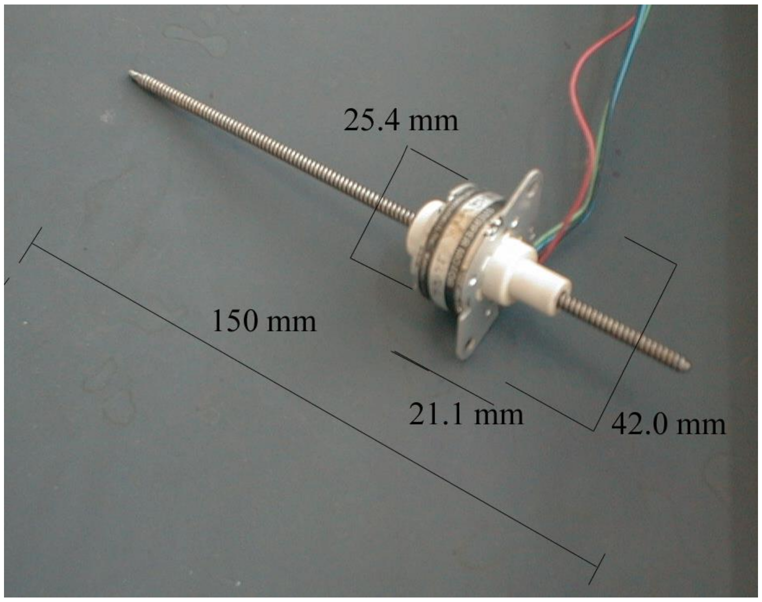


Figure 9. Picture of stepper motor used with dimensions added.

VI.3.2 Microsyringe info and syringe used

The syringe was a key component that varied as the system progressed. All syringes used were from Hamilton Company (Reno, NV). The two key parameters of the syringes that were varied were the type of syringe (Gastight or Microliter Model) and the syringe volume. The type of syringe mainly contributed to the resistance the plunger offered back to the motor. The syringe volume determined the amount of fluid deposited for a given displacement of the plunger. Smaller volumes are better for smooth fluid deposition, as multiple passes can even be taken to smooth out the fluid. However, the smaller the syringe volume, the more often it needs to be refilled, which is an inconvenience to the user. The bubbling problem was also more prevalent with the smaller syringes. The final syringe chosen was the 250 µL Microliter syringe. This syringe gave reliable fluid deposition, and refilling the syringe was determined not to be a problem. The total syringe volume is closer to 300 µL, and this allows the user to make up to 320 membranes between refills.

VI.3.3 Needles

The needle is a key system parameter because it is the closest part of the system to the membrane. The needle is also the source of the largest problem encountered, that of the solution bubbling up on the end of it. Since the fluid is being pushed from the top with a constant volumetric velocity, a smaller needle (larger gauge) will cause the fluid at the tip to be ejected with a larger linear velocity. A removable needle is also preferred to a fixed one. It can be difficult to gain a good seal with a removable needle, especially the Luer Tip (LT) syringe used. However, since plungers and barrels are not interchangeable, breaking or clogging a needle would require the user to order and modify an entire new syringe. Experiments were conducted to assess the efficacy of different needle types.

Table 3- Various needles used and their suitability for the desired task. The table lists the three needles used and various parameters relating to their operation.

Needle Used	Gauge	Wall Thickness [mm]	Dead Volume [µL/cm]	Comments
91033	33	0.05	0.08	Needle proved to be significantly too small. It was difficult to draw up fluid and the needle would bend as the print head was moving laterally.
91025	25	0.13	0.49	Needle worked well. The wall thickness was significant enough to ensure stability.
Default Needle for 750SNR 500 µL Syringe	23	0.15	0.85	This needle worked fairly well, but the large volume contributed to bubbling. This needle was also not removable.

The final system uses the 91025 needle because of its good performance. A good seal can be obtained by tightly wrapping a parafilm coating around the area where the syringe barrel and needle intersect.

VI.3.4 Machine shop design

The fluid deposition system started out with two rectangular pieces of translucent Plexiglas plastic. A standard mill and lathe were used to cut several channels to house the syringe and motor. Notches were cut into the ends so that the two pieces could fit together. Finally, 0.3 cm diameter holes were drilled into the motor housing unit to hold the stepper motor in place and screws were inserted through the motor flanges and into the syringe housing unit. (Figure 6) Aluminum pieces were bent and had rubber cemented onto them to hold the syringe in place. (Figures 7-8)

VI.3.5 Motor/plunger interface

The most difficult part of the fluid deposition system design was determining how to attach the bottom of the motor screw to the top of the plunger. Fortunately, the bottom of the motor screw had a standard size to allow for a smooth connection. The motor had a #4-40 UNC-2A thread measuring 3/56''. There were two key criteria for choosing a method to attach the screw and plunger. Most importantly, the interface should be as straight as possible. Any deviations would cause a lopsided movement of the plunger down which would result in a periodic amount of fluid being dispensed for a constant amount of motor steps signaled from the control program. Less importantly, the interface should be removable to aid in cleaning the syringe or in changing a broken syringe, since Hamilton Microliter Syringe plungers are not interchangeable with different barrels of the same model. A third desired criterion would be that the interface be capable of bi-directional motion. This could possibly allow for automatic syringe refill controlled through the computer. However, due to the difficulties with a bi-directional attachment a unidirectional device was constructed.

Three separate designs were made in the machine shop for three different syringe plungers. Option A uses a direct attachment method, while options B and C use indirect attachment. Figure 10 shows an image of the attachment methods.

The first design (option A) consisted of drilling a hole directly in the top of the 500 μL syringe plunger. The hole was then tapped and threaded at 3/56'' using standard techniques. The syringe plunger top was hollow, so the hole was drilled directly in the plunger. This direct method of attachment worked well, but was difficult to achieve and required the modification for every syringe. It would also not work on smaller syringes with thinner plungers.

The second design (option B) involved taking a 1 cm cylinder of aluminum and drilling, taping and threading the hole in this piece only. This modification was selected for the 250 μL gastight syringe which was found to have a solid plunger head. The curved top of the solid plunger head was shaved off and the cylinder was attached with epoxy. Now, if a new syringe were required, the epoxy could be dissolved or melted by heating it and the cylinder could be attached to the next syringe. However, lining up all the pieces was still a problem, and although now only one small lathe step was required, the connection still could not be made directly.

The last and most successful design (option C) involved a larger cylinder that had a spherical indentation removed so that it fit directly over the plunger head. The attachment port for the motor screw is on the other side. Now, the syringe plunger could be modified directly out of the box, and it was easier to achieve the correct alignment. The direct modification was important because the machine shop steps could damage the delicate syringe or the plunger. The plunger was so precisely machined for its particular syringe barrel, even small depressions or projections on its surface from clamps could hinder its movement in the barrel, significantly increasing the force required to travel a certain distance in a non uniform manner that was deleterious to the repeatability of the overall device.

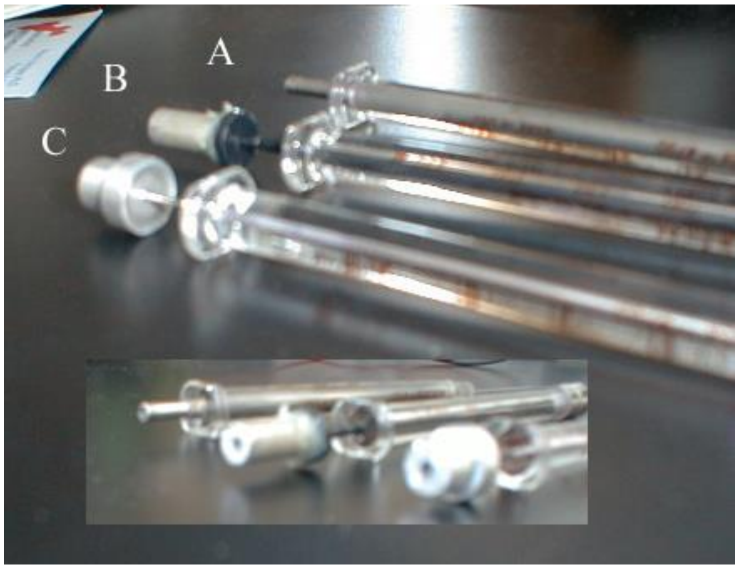


Figure 10. Syringe attachment options. The inset image shows option A on the left, option B in the middle, and option C on the right. Option C was chosen because it requires no destructive modification to the plunger and it provides excellent alignment.

VI.3.6 Bubbling problem

One of the most challenging aspects of the device design was the problem that the volume of fluid ejected from the syringe in a specific period of time did not equal the fluid deposited on the membrane in that same period of time.

The fluid ejected tended to ‘bubble up’ on the point of the needle due to the high surface tension of water. For very small fluid speeds, the bubble increased until the acceleration due to gravity overcame the surface tension and the bubble fell and splashed onto the membrane. At faster fluid speeds, the kinetic energy overpowered surface tension and the bubbling problem disappeared. Thus, a smaller gauge needle was used to increase the velocity of the fluid ejected. Also, the use of other point styles of needles to bring in air or disrupt the bubbling somehow was briefly investigated, with a representative from Hamilton confirming that a blunt tip that perpendicular to the length of the needle (point style three) was the best.[18] The various needle point styles are depicted in figure 11.

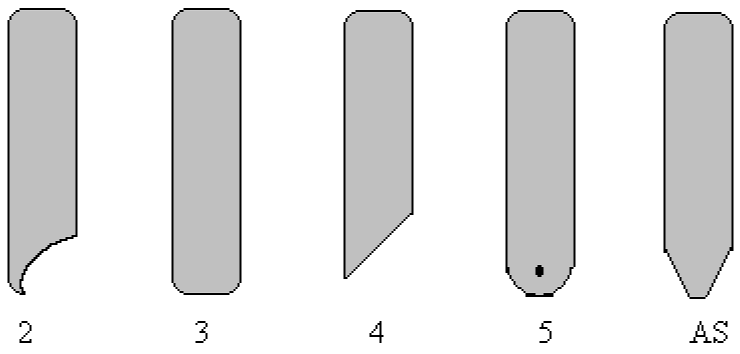


Figure 11- Possible needle point styles. The following needle point styles are available from Hamilton. Point Style 2 has a beveled non-coring point and is recommended for septum penetration. Point style 3 is used in HPLC injection valves. Point style 4 has a 10°-12° beveled needle point and is used in life sciences Point Style 5 is a Conical needle with side port used in penetration of septa, thin-gauged vinyls and plastics. Point Style AS is used in autosamplers.[17]

The current method of pipetting the solution onto the membrane relies on the wicking action of the membrane to pull the solution out of the tip. This same phenomenon can be used in the automated system if the syringe is kept close enough to the membrane. However, keeping the needle close to the paper is difficult, and if the needle is too close it can scratch the membrane. Experiments with the 50 µL syringe confirmed that it was the volumetric fluid speed that was more significant in determining whether bubbling occurred than the linear speed of a particular fluid molecule. However, if the fluid speed is too fast, then splashing can occur. Splashing would ruin the lines of fluid and is not desirable. The Weber number (We) gives the propensity of a given fluid to splash, and it is the ratio of fluid inertial to membrane capillary effects. The Weber number is given by the equation:

$$We=\rho r v^2/\gamma$$

Where ρ is the fluid density, r the radius of the fluid drop, v the impact velocity, and γ the surface tension of the drop. Splashing is thought to occur at Weber numbers > 50.[19] However, this analysis is approximate because the Weber number is only for a droplet of fluid, not a continuous stream. The needle tip was placed approximately 1 mm from the membrane.

VI.3.7 Syringe Refill

A detailed protocol for refilling the syringe was developed. A key feature of this method is that the syringe does not need to be moved. This way, the careful alignment of the syringe to the membrane may be preserved. The basic idea involved removal of many components on the right side of the printer with a hacksaw so that the fluid in a beaker or microcentrifuge tube may be brought in by hand from the right. Now, the microcentrifuge tube or small beaker containing the desired fluid can be brought up under the syringe. The user then draws the fluid up into the syringe by turning the motor shaft/plunger interface counterclockwise.

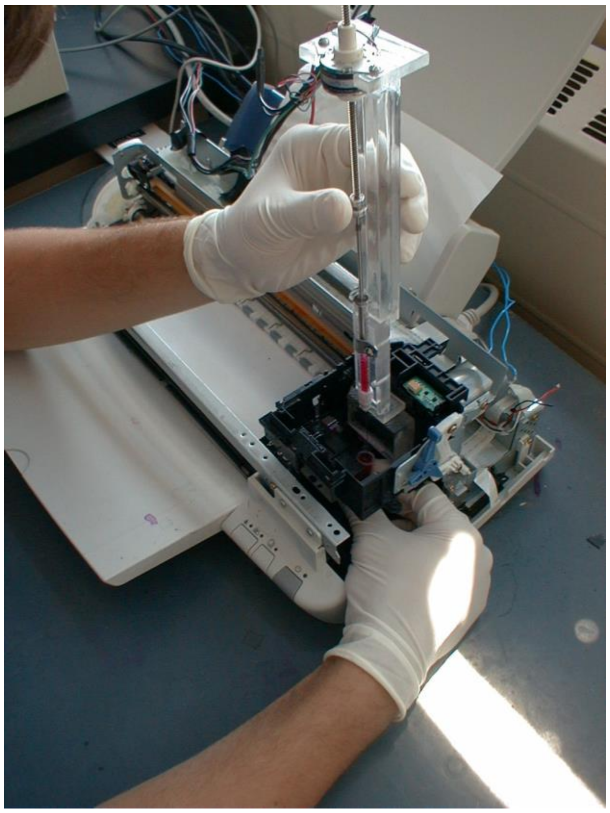


Figure 12- Technician refilling the syringe. The eppendorf tube containing the desired liquid is inserted into the system through the hole in the right side of the device. The liquid has been stained red with sulfhrodamine-B dye.

VI.3.8 Cleaning

A detailed cleaning protocol that also preserves the position of the syringe has also been developed. It is important to clean the system to avoid proteins getting stuck in the small needle. The needle can be removed and sonicated for 30 seconds at room temperature with a degassing level of 9. The syringe itself can be cleaned by flushing repeatedly with deionized water. This should be done after each use for the day, or when changing solutions. Periodically the other mechanical components may be cleaned and oiled if necessary.

VI.4 Computer Interface/Software

VI.4.1 General digital I/O and labPC info

Digital input and output is a key aspect of any electrical or mechanical system that operates in the physical world but is controlled by a software program. Ever since the first computers were developed, engineers have worked on digital input and output systems so that the computers could interact with objects in the physical world. Since the development of personal computers in the 1980s, digital input/output systems are commonly found on removable PC cards. Such systems are now in wide commercial production. This application uses a National Instruments NI-DAQ 6503 PC Board. The current system requires only six ports of digital output that can be switched on the order of milliseconds. While it would greatly simplify the electrical control circuitry to have a PC board that could provide 0 to 24 VDC at up to one ampere, this is not feasible, as the boards are not designed to power any significant appliances. Instead, the board is used only as a control to direct the flow of current from a larger external power supply. The 6503 board can handle up to 8 bit digital output on 3 different ports for a total of 24 control ports. National Instruments products are also attractive because they include the National Instruments Digital Input/Output (NIDAQ) library of C++ header files and functions to control their devices. There is a wide variety of digital I/O systems but the 6503 board was selected because it was the most inexpensive model available.

VI.4.2 Control algorithm and modularization design

Sophisticated software packages to control digital I/O boards exist, most notably the application package Labview from National Instruments. However, due to the high cost of the software and the relative simplicity of the control program, a stand-alone application was built in C++ compiled with Microsoft Visual C++ version 6.0. C++ is attractive because with the NIDAQ library fast programs can be created that allow the programmer control over individual output bits by writing a specific value to the output registers. For example, to turn on (set to one) port A positions one and four the programmer can write a binary twelve (00010010b) to port A. Table 3 lists the main functions and classes created.

Table 4- C++ classes written or modified for use in the MembranePrinter.exe application. The following table details the main functions used. Functions in standard libraries that were imported and not changed are not listed.

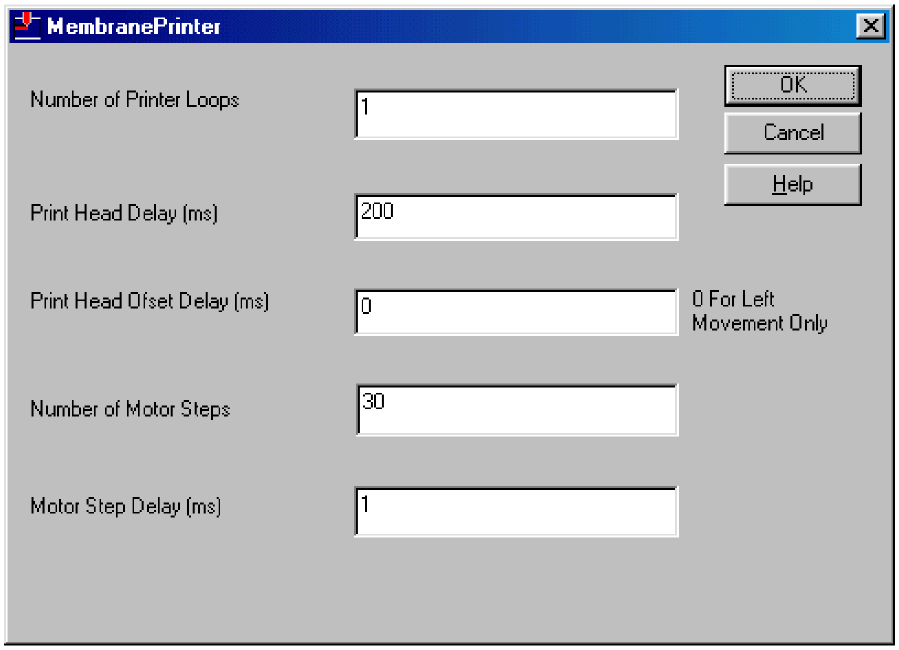
Name	Type	Description	Parameters
CMembranePrinterApp	Class	The main class that contains the application	None
CMembranePrinterDlg	Class	Class that makes the dialog box	Pointer to the window
CaboutDlg	Class	Makes about box	None
Printercontroller	Class	Main class that controls the system movement	None
delay()	Function	Delays dt milliseconds	int dt
motorUP()	Function	Puts the motor up a specified number of steps using the stepDelay	int numSteps
motorDown()	Function	Puts the motor down a specified number of steps using the stepDelay	int numSteps
setValues()	Function	Sets the five key parameters in the printercontroller object	int: NumLoops headDelay numSteps stepDelay ofsetDelay

The software is designed in a modular fashion using C++ class objects to facilitate ease of design and testing. The main function is contained within the MembranePrinter class which starts up the dialog based GUI. The dialog takes in the user selected run parameters and sends them to the main printercontroller object. The printer controller object contains functions like motorUP() and motorDown() to control the motion of the stepper motor. The main function of the printercontroller object is the run() function, which loops through all of the control commands in order and separated by the correct delay. Both motors need to run simultaneously. One way to do this is to have the program run with threaded execution with each motor being controlled by an independent thread. However, a simpler solution is to set the print head motor to move in a certain direction, then complete the motion of the stepper motor, and then delay a user-specified time before turning the print head motor in the opposite direction. The delay time must always be positive or zero because the print head must run for at least as long as it takes to deposit fluid onto the membrane. The following pseudocode details the algorithm with the key parameters. Curly brackets denote loops of execution.

```
Do numLoops times {
    Set printer to move left (write 01 to port A)
    Do numSteps times {
        Step through (1001-0101-0110-1010) with a stepDelay each time
    }
    Delay headDelay
    If delayOfset !=0
        Set printer to move right (write 10 to port A)
    Delay headDelay+delayOfset
}
```

VI.4.3 Graphical User Interface

The graphical user interface allows users easy access to the various operating parameters that they need to control. It is familiar and easy to use for technicians used to the windows operating system environment. Figure 13 shows the application's main dialog box.

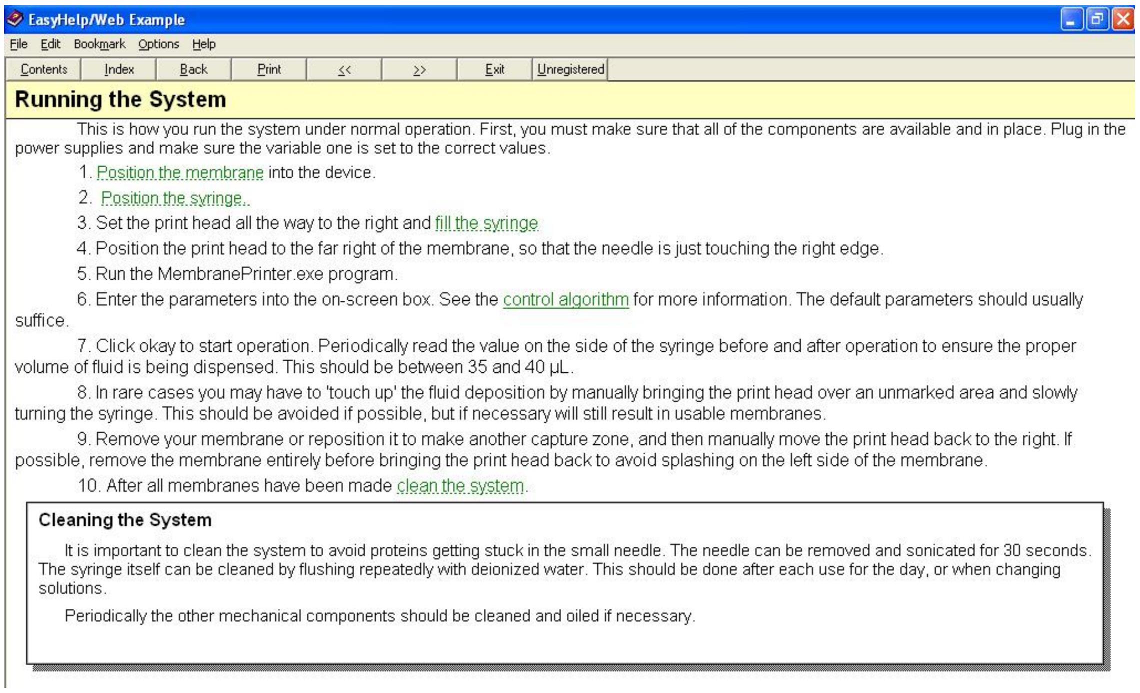


Number of Printer Loops	1	
Print Head Delay (ms)	200	
Print Head Ofset Delay (ms)	0	0 For Left Movement Only
Number of Motor Steps	30	
Motor Step Delay (ms)	1	

Figure 13- Screen capture of the main dialog box of the MembranePrinter application. The user simply needs to enter the appropriate values and click okay. Cancel will stop program execution and return the user to windows. A detailed help file is also available to users clicking on the help button.

VI.4.4 Windows Help File

An extensive windows help file was also created to provide directions and support to the user during program operation. The file was compiled with a shareware copy of EasyHelp from Eon Solutions. (Macclesfield, England) Separate help contents and help project files were created and subsequently linked into the Visual C++ compiler to create the final document. Larger topics such as the table of contents, running the system, common problems, important reminders, and the control algorithm are arranged as jump topics, while other topics are arranged as popup topics. This allows the user quick access to small pieces of information peripheral to a larger task. Figure 14 shows a screenshot from one part of the help file.


EasyHelp/Web Example
File Edit Bookmark Options Help
Contents Index Back Print << >> Exit Unregistered

Running the System

This is how you run the system under normal operation. First, you must make sure that all of the components are available and in place. Plug in the power supplies and make sure the variable one is set to the correct values.

1. Position the membrane into the device.
2. Position the syringe.
3. Set the print head all the way to the right and fill the syringe
4. Position the print head to the far right of the membrane, so that the needle is just touching the right edge.
5. Run the MembranePrinter.exe program.
6. Enter the parameters into the on-screen box. See the control algorithm for more information. The default parameters should usually suffice.
7. Click okay to start operation. Periodically read the value on the side of the syringe before and after operation to ensure the proper volume of fluid is being dispensed. This should be between 35 and 40 µL.
8. In rare cases you may have to 'touch up' the fluid deposition by manually bringing the print head over an unmarked area and slowly turning the syringe. This should be avoided if possible, but if necessary will still result in usable membranes.
9. Remove your membrane or reposition it to make another capture zone, and then manually move the print head back to the right. If possible, remove the membrane entirely before bringing the print head back to avoid splashing on the left side of the membrane.
10. After all membranes have been made clean the system.

Cleaning the System

It is important to clean the system to avoid proteins getting stuck in the small needle. The needle can be removed and sonicated for 30 seconds. The syringe itself can be cleaned by flushing repeatedly with deionized water. This should be done after each use for the day, or when changing solutions.

Periodically the other mechanical components should be cleaned and oiled if necessary.

Figure 14- Screen shot from the windows help file. When the user clicks on one of the underlined topics, a popup window comes up offering more information. This screen shot would occur when the user clicked on the "clean the system" text.

VI.5 Electronic Control Circuitry

VI.5.1 Circuit diagram and control system information

The main control circuit was constructed on a breadboard used for prototyping. The original design consisted of an array of transistors used as on-off switches to control the motors. However, due to the difficulty in designing this setup for bi-directional motor movement and stepper motor control, the current system using 293 D H bridge chips from Thomson Microelectronics and Texas Instruments was adopted. The current system is also more robust and well characterized. A circuit diagram of the chip interconnects is shown in figure 15.

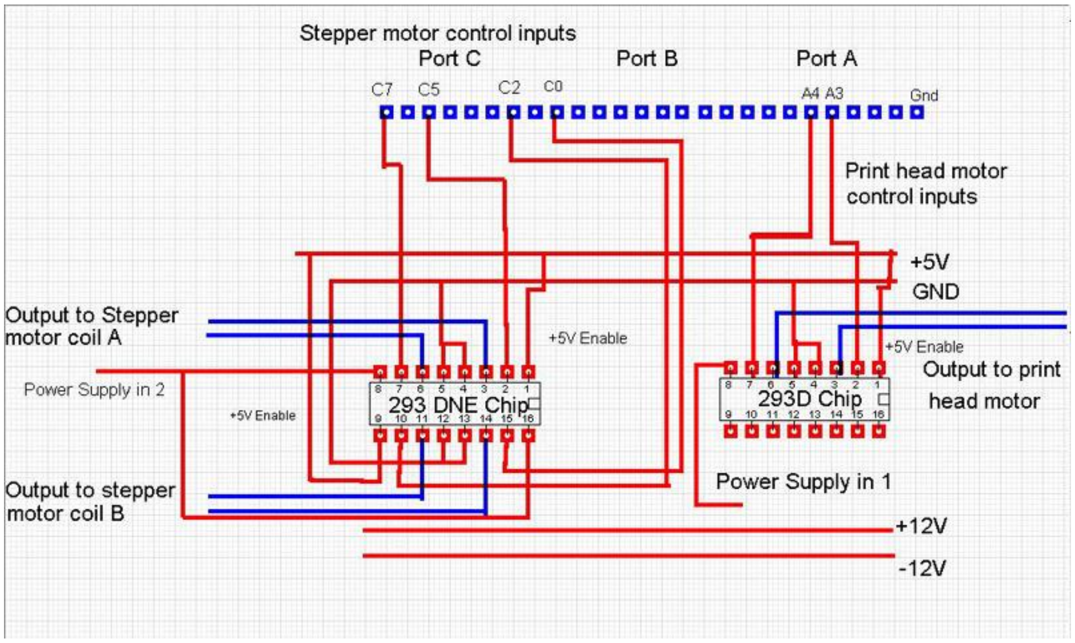

Stepper motor control inputs
Port C
Port B
Port A
C7
C5
C2
C0
A4 A3
Gnd
Print head motor control inputs
+5V
GND
Output to Stepper motor coil A
+5V Enable
+5V Enable
Output to print head motor
Power Supply in 2
+5V Enable
293 DNE Chip
293D Chip
Output to stepper motor coil B
Power Supply in 1
+12V
-12V

Figure 15- The main control circuit. The main control circuit is on a breadboard and consists of two 293D H Bridge driver chips. One chip is dedicated to controlling both stepper motor heads, while the half of the other chip is used to control the print head motor.

VI.5.2 Power supply

The system actually uses three separate sources of power. Control signals come in from the National Instruments board. Chip enable signals come in at +5 V from the breadboard power supply. The main power comes from a variable two channel external supply. This is necessary because neither of the other two supplies can provide high enough voltages or currents. The external supply was manufactured by Trygon (Westbury, NY) and provides 0-40 V at up to 1 A.

VI.5.3 System operating parameters

The system operates under a defined set of parameters. Operating outside of this range can cause damage to the system or result in unstable fluid deposition. For example, operating under high voltages or with extremely long delays (on the order of hours or days) can cause the system to overheat and burn out the chips or the motors. Table 5 details the parameters and their normal and absolute maximum values.

Table 5. Range of system operating parameters. The parameters come from the specification sheets of individual components as well as overall system testing.

Parameter	**Minimum**	**Normal Maximum**	**Absolute Maximum**
Print Head Voltage	0 V	15 V	36 V
Stepper Motor Voltage	0 V	20 V	36 V
Enable Voltage	4 V	5 V	36 V
Print Head Speed	0 cm/s	20 cm/s	30 cm/s
Current	0 A	0.25 A	1.0 A
Temperature	0^o C	30^o C	50^o C
Step Delay	1	2	4

VII. Testing and Results

VII.1 Figures of merit

The device was tested with respect to the several important figures of merit mentioned above.

<u>Ease of Use:</u> The device seems easy for a lab technician to operate with only minimal training. Detailed protocols and help documentation have been established to guide users through the entire process.

Non-Interference with Biological Processes- There has not been any noted interference problem. However, the system should be kept as clean as possible to avoid possible nuclease contamination.

Repeatability: The system has a high level of repeatability. There are three important measures of the repeatability of the device. First, the amount of fluid ejected from the syringe should be a linear function of the number of steps input. Figure 16 shows a graph obtained with a 14 V print head motor voltage, 20 V stepper motor voltage, 1000 ms head delay, 0 ms offset delay, and 1-4 ms step delay. Note the high value for R^2 (0.9914)

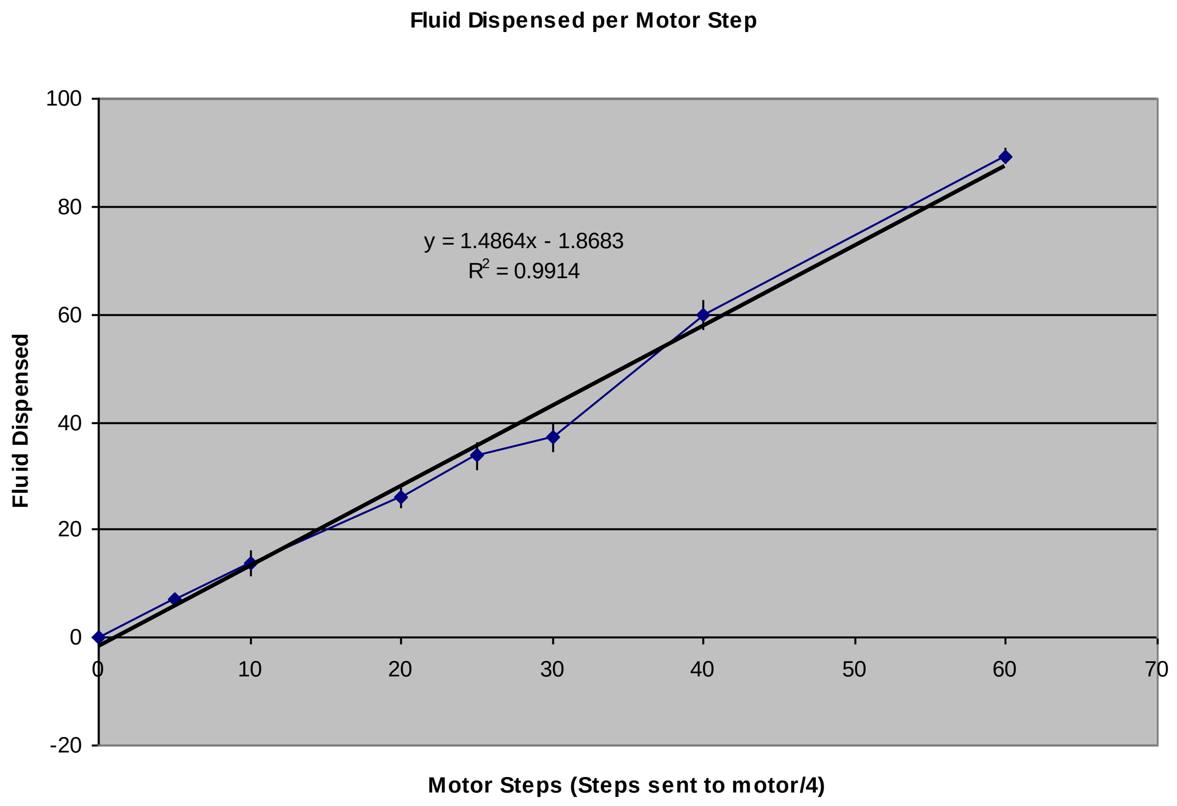


Figure 16. The amount of fluid dispensed per motor step. This graph was obtained with a 14 V print head motor voltage, 20 V stepper motor voltage, 1000 ms head delay, 0 ms offset delay, and a 1-4 ms step delay. The error bars represent one standard deviation. This is only the fluid ejected from the syringe, and not necessarily the fluid deposited to the membrane. For each step input to software, the motor actually steps through one sequence which is equivalent to 4 motor steps.

For the default parameters, this leads to a deposition of an average of 37.1 µL per sweep, with a standard deviation of 2.83 µL. The next important parameter that must be repeatable is the size of the lines. The lines should not deviate from the average value. They should also be as straight as possible. Figure 17 shows a typical set of lines produced by the system laying down a solution of Bovine Serum Albumin (BSA) and Sulfrhodamine B dye on the polyethersulfone membrane.

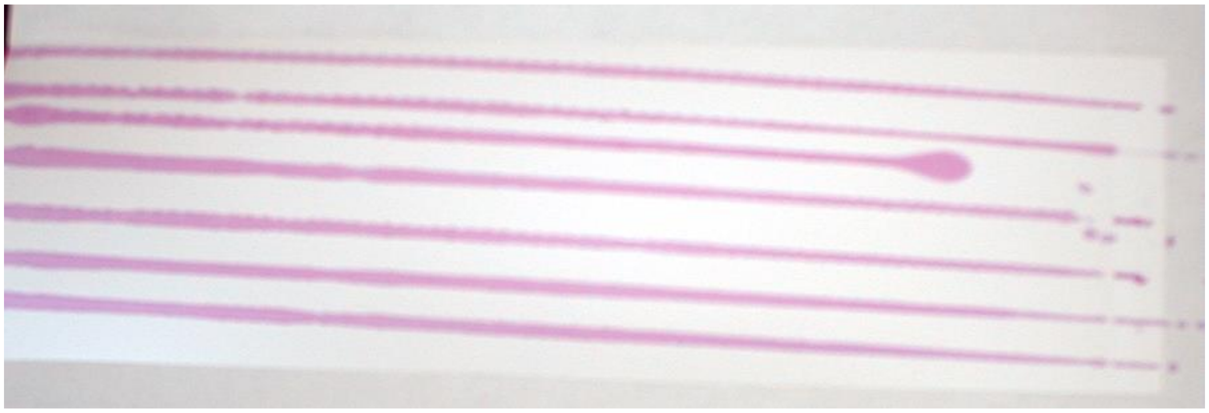


Figure 17. A typical example of the fluid deposition capabilities of the system. Note that the inconsistencies on the far right side are caused by the tape attaching the membrane to the paper. This section plus one cm is not used. Also, the bubble in the third line from the top is present to show what can happen if there is a bubble in the needle before deposition begins.

The width of the lines was measured in one cm intervals along the 18 cm line to obtain a histogram of line widths. Figure 18 depicts a histogram showing how many points had a specified width. The average line width was 1.91 mm and the standard deviation was 0.54 mm. The desired line width is 2 mm, as this is the diameter of the reflectometer opening. The standard deviation is higher than desired, but should not affect the overall signal significantly, as the number of available capture probes should be larger than the number of target sequence/liposome complexes, even at the lower end of the fluid line width.

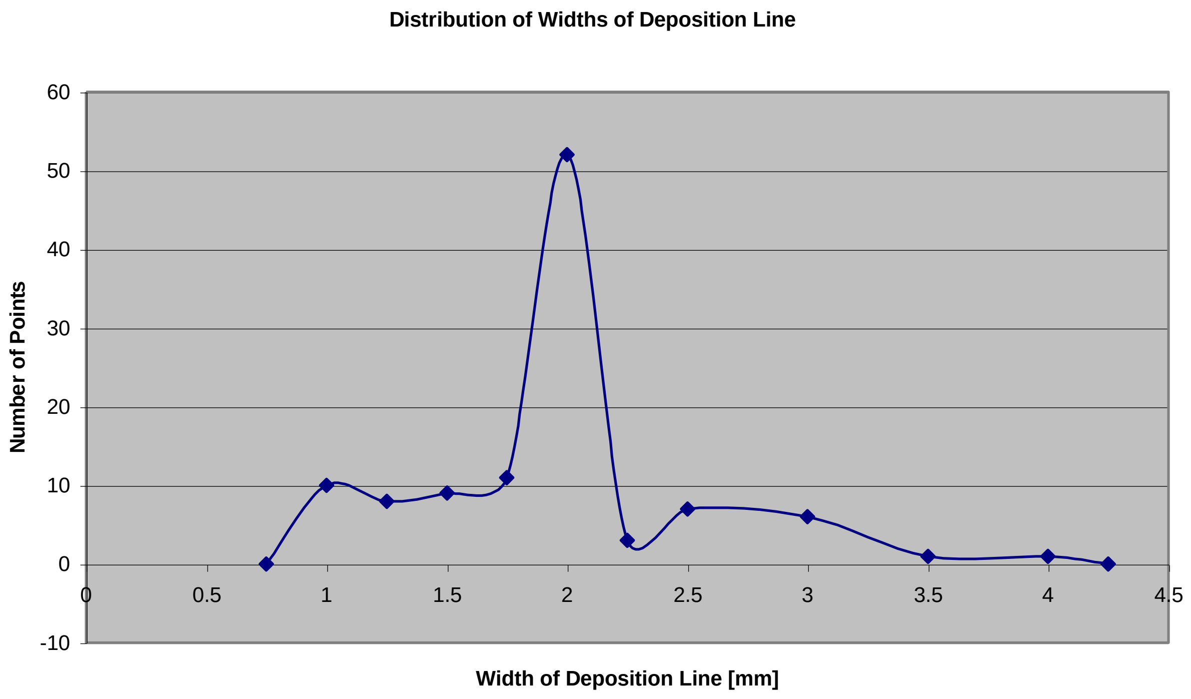


Figure 18. Histogram depicting the width of the fluid line in mm at specified points along the line. 108 points were used to obtain this distribution. The average value is at 1.91 mm and the standard deviation is at 0.54 mm.

The last and most repeatability factor to consider is the standard deviation of the reflectometer signals that the membrane provides. However, this value is a function of many more parameters that have to do with making solutions and running assays. For a detailed discussion of this factor see the section on comparison to other methods.

VII.2 Factors to improve performance

There are many factors that can improve or impair the performance of the system. Small deviations on some parameters can lead to large problems. If the needle is dirty or clogged, then the fluid will come out unevenly. The state of the Windows environment is also important. If other programs are running or shared files on the computer are being accessed remotely, the software timing functions can give non-uniform results. If the computer has a long uptime, it may be necessary to restart it before running the system. Accordingly, if the system has not been used recently, it may be beneficial to lay down a practice line of water or buffer solution on the paper in front of the membrane to ensure that the system is working correctly. However, if the correct protocols are followed for positioning the membrane, positioning and refilling the syringe, and running the system are followed, the device will give reliable and repeatable operation.

VIII. Device Operation

VIII.1 Default parameters

The device operation has been optimized and several default parameters have been specified. In general, these values should not need to be changed often. However, a change in computing environment (new computer or operating system, more memory, etc) could cause changes that would require the system to be recalibrated.

Table 6- Default system operating parameters. The following table lists the important system variables, their default (optimized) value, and the sensitivity of the system to changes in that value.

Parameter	Default Value	Sensitivity
Number of Loops	1	High- running a different number of loops will greatly change the system. Running x more loops will cause x times more fluid to be ejected.
Print Head Delay	200 ms +/- 150 ms	Low- As long as this value is greater than zero the system should function correctly, as fluid is being deposited until the edge of the membrane.
Ofset Delay	0	Low- This value is set to zero by default to flag that the print head should only move left.
Number of Steps	30 +/- 2 steps	Medium- This parameter will determine the amount of fluid ejected, but unless the print head speed is slow enough the fluid may not all go onto the membrane.
Step Delay	1 ms	Medium- Values higher then 4 ms will start to give dotted membranes.
Print Head Voltage	15 V +/- 1V	High- Values lower than 7 V will cause the print head to not move smoothly, values less than 14 V will affect the smoothness of the lines on the membrane.
Stepper Motor Voltage	20 V +/- 1V	Medium- Changing this value will change the thrust of the motor and could cause a different amount of fluid to be deposited.

VIII.2 Performance outside designated range

In general, the system has not been thoroughly tested outside of the designated operating range. However, the system should perform roughly as outlined in figure 16. The largest amount of fluid that can be deposited is 300 µL, and the smallest bubble that can fall to the surface is 3 µL. At these depositions, however, there is no guarantee that the fluid will be evenly deposited on the membrane. Depositions above 40 µL will require either looped operation or multiple one-loop depositions on the same spot in the membrane.

VIII.3 Comparison to current methods

In general, the system performs well when compared to current techniques. 148 membranes of each type were made with three capture zones to test for antibiotic resistance genes.[20] The membranes made with the automated system have signals in bands rather than the curves of the membranes made by hand. This is advantageous because it allows the user to read the signals more easily. It also opens up the possibility of automating the data collection. Figure 19 shows a comparison between signals made with both techniques.

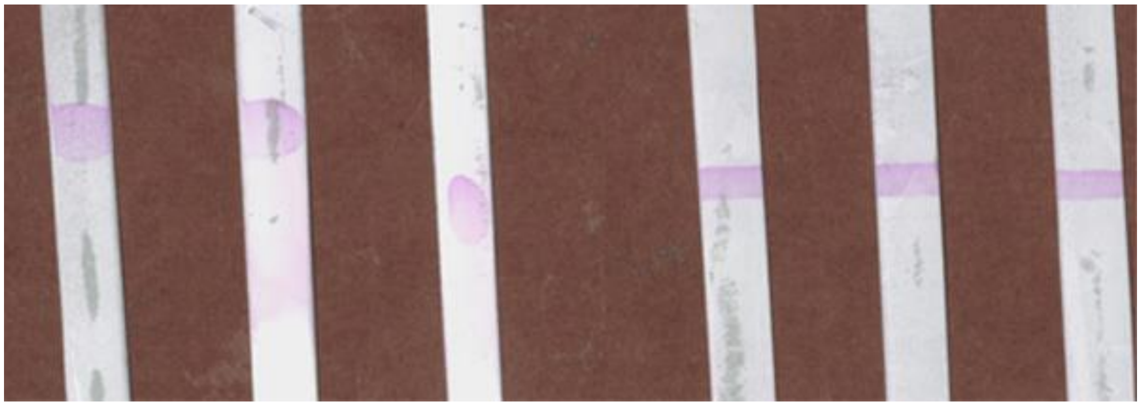


Figure 19- Comparison between signal of membranes made with standard techniques (left) and membranes made with the new system (right). The membranes made with the new system have smaller signal bands in a rectangular pattern. This allows the technician to get more of the signal inside the reflectometer reading hole. The assay was run with 5000 fmol/µL of Kanamycin resistance gene target analyte.

Another important advantage of the automated system is that since the signal is more concentrated, a larger amount of immobilized dye can be placed within the reflectometer sample window. This results in approximately a 35% higher signal for membranes made with the automated system. The membranes made with the automated system also have a higher concentration of immobilized capture probe. This is necessary because of the slightly smaller amount of fluid being deposited to the membrane. The amount of capture probe is the same for both membranes. Figure 20 shows standard curves generated for both sets of membranes.

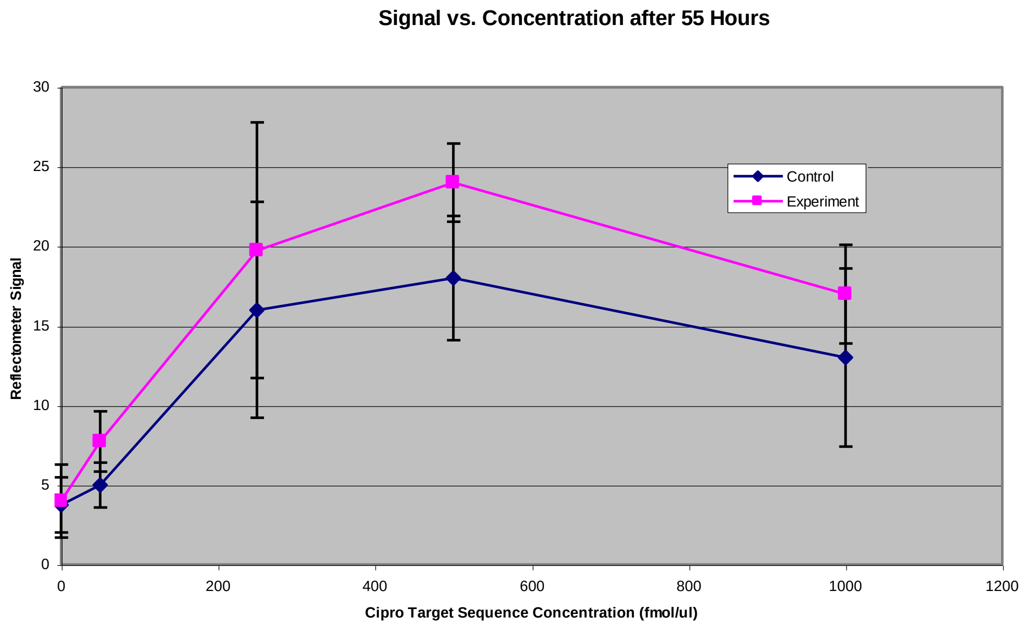


Figure 20- Signal versus concentration for control (bottom curve) and experimental (top curve) membranes. The most important thing to note is that the shapes of the curves are similar, and that the deviations of the experimental curve are smaller than those of the control curve.

It is important to note that the shapes of the curves are similar. Also, the standard deviation around the points made with the automated system (experiment) is on average 7% less than the standard deviation around the points made with traditional techniques (control). The experiment curve is shifted vertically. This is because the signal is on the average 35% higher. This should lead to biosensors with a lower value for their limit of detection and a larger dynamic range.

Non-specific binding does not seem to be a problem with either set of membranes. The membranes have three different capture zones, yet the signal is seen only in the appropriate capture zone. Negative controls also reveal a low and uniform background signal.

The membranes made by the automated system also save the user a significant amount of time. The various tasks necessary in making the membranes were timed and table 7 lists the approximate time for each task. This chart was made for 148 membranes made each way. The actual time saved will depend on many factors, including the experience of the particular person making the membranes. It is hoped that the automated system will reduce this variability. The amount of time saved should be roughly a linear function of the number of membranes made, with all automation advantages disappearing at only one membrane being made.

Table 7. Time saved by automated system. The following chart depicts the approximate time for various tasks for both the standard and automated system. Each protocol made 148 membranes.

	Time to Complete (Technician Minutes)	
Task	**Standard Techniques (Control)**	**Automated System (Experiment)**
Cut membranes	60	0
Mix probes/ streptavidin solution	10	10
Position large membrane on paper	0	10
Set up cardboard and position small membranes	50	0
Deposit fluid on membranes	75	27
Dry in oven	90	90
Transfer to blocking solution	25	5
Blocking	30	30
Transfer to Kimwipes	12	1
Transfer back to carbdoard and place in hood	50	2
Dry in Oven	150	150
Place in Vacuum Bags	30	30
Cut Membranes	0	60
Total Time	582	415
Operator Time	312	145
Absolute Reduction in Total Time	167	
Relative Reduction in Total Time	29%	
Relative Reduction in Tech Time	54%	

Because there are many long periods of waiting in the protocol, the relative reduction in researcher time is much more significant than the relative reduction in absolute time. During these waiting times, the technician may go on to other tasks.

The device compares less favorably with commercially available devices, such as the Linomat 5 and ATS4. These devices were designed for a large range of tasks, and are much more flexible than the current system. They also are better in terms of repeatability and precision of fluid deposition, and have more extensive control systems. However, they cost many thousands of dollars more than the described device.

IX. Conclusion

A modified membrane making protocol that includes the automated system to deposit fluid in one line can significantly improve the quality of membranes produced. The modified protocol also saves technician time and is easier for new users to perform. The device described has some small problems, most notably with repeatability of line width and inability to deposit an arbitrary amount of fluid. However, none of these problems preclude the use of the device in a normal laboratory setting. For users who want to deposit a line of fluid, this device should prove a reliable and inexpensive alternative to commercially available systems.

X. Appendix

X.1 Specification Sheets for chips and NIDAQ board

Following are the specification sheets and key technical information for the 293 H bridge chips, stepper motor, and the National Instruments Board.

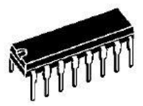
SGS-THOMSON MICROELECTRONICS

L293D
L293DD

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

- 600mA OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (non repetitive) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES

SO(12+4+4)


Powerdip (12+2+2)

ORDERING NUMBERS:

L293DD	L293D

DESCRIPTION

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoides, DC and stepping motors) and switching power transistors.

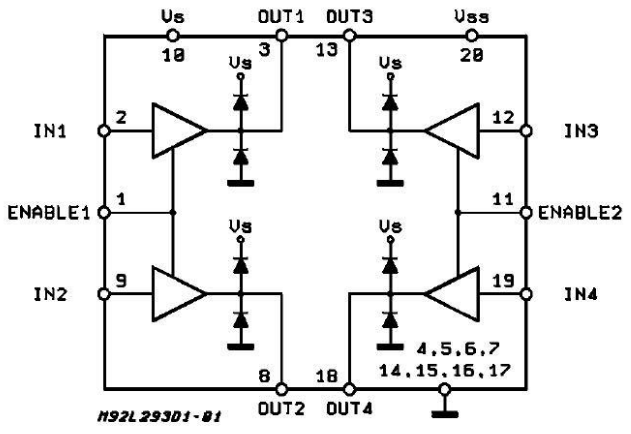
To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz.

The L293D is assembled in a 16 lead plastic packaage which has 4 center pins connected together and used for heatsinking

The L293DD is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heatsinking.

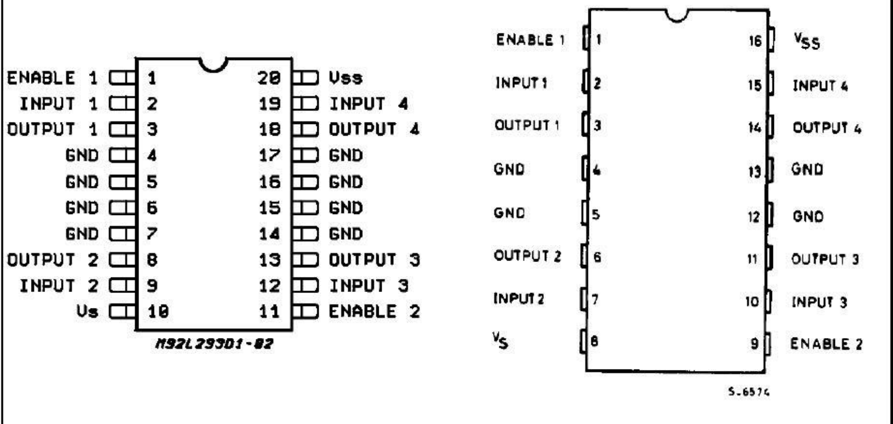
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{en}	Enable Voltage	7	V
I_o	Peak Output Current (100 µs non repetitive)	1.2	A
P_{tot}	Total Power Dissipation at T_{pins} = 90 °C	4	W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to 150	°C

PIN CONNECTIONS (Top view)



SO(12+4+4)

Powerdip(12+2+2)

THERMAL DATA

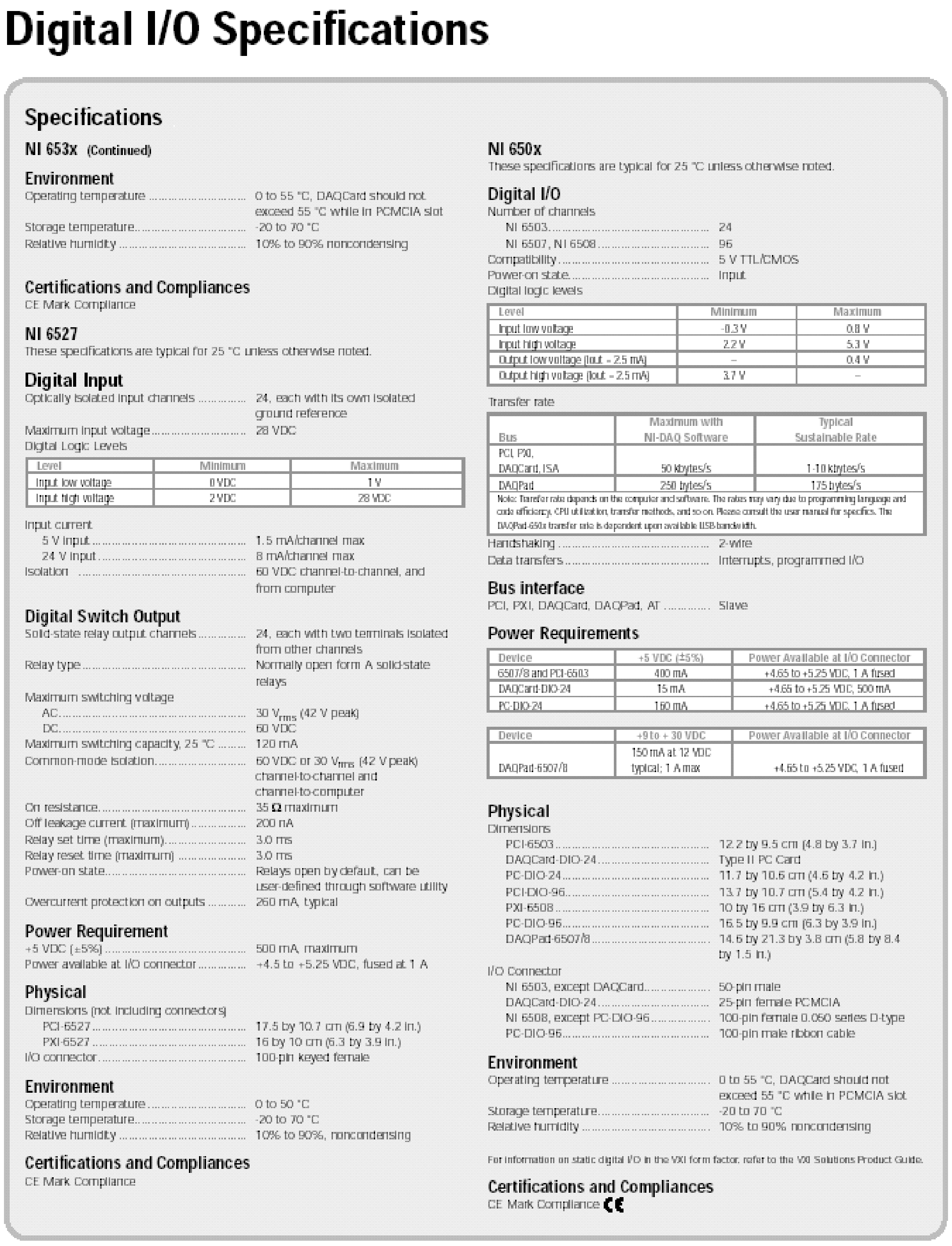
Symbol	Decription		DIP	SO	Unit
$R_{th\ j\text{-}pins}$	Thermal Resistance Junction-pins	max.	–	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance junction-ambient	max.	80	50 (*)	°C/W
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	max.	14	–	

(*) With 6sq. cm on board heatsink.

ELECTRICAL CHARACTERISTICS (for each channel, V_S = 24 V, V_{SS} = 5 V, T_{amb} = 25 °C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 10)		V_{SS}		36	V
V_{SS}	Logic Supply Voltage (pin 20)		4.5		36	V
I_S	Total Quiescent Supply Current (pin 10)	V_i = L ; I_O = 0 ; V_{en} = H		2	6	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	24	mA
		V_{en} = L			4	mA
I_{SS}	Total Quiescent Logic Supply Current (pin 20)	V_i = L ; I_O = 0 ; V_{en} = H		44	60	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	22	mA
		V_{en} = L		16	24	mA
V_{IL}	Input Low Voltage (pin 2, 9, 12, 19)		– 0.3		1.5	V
V_{IH}	Input High Voltage (pin 2, 9, 12, 19)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
I_{IL}	Low Voltage Input Current (pin 2, 9, 12, 19)	V_{IL} = 1.5 V			– 10	µA
I_{IH}	High Voltage Input Current (pin 2, 9, 12, 19)	2.3 V $\leq V_{IH} \leq V_{SS}$ – 0.6 V		30	100	µA
$V_{en\ L}$	Enable Low Voltage (pin 1, 11)		– 0.3		1.5	V
$V_{en\ H}$	Enable High Voltage (pin 1, 11)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
$I_{en\ L}$	Low Voltage Enable Current (pin 1, 11)	$V_{en\ L}$ = 1.5 V		– 30	– 100	µA
$I_{en\ H}$	High Voltage Enable Current (pin 1, 11)	2.3 V $\leq V_{en\ H} \leq V_{SS}$ – 0.6 V			± 10	µA
$V_{CE(sat)H}$	Source Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = – 0.6 A		1.4	1.8	V
$V_{CE(sat)L}$	Sink Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = + 0.6 A		1.2	1.8	V
V_F	Clamp Diode Forward Voltage	I_O = 600nA		1.3		V
t_r	Rise Time (*)	0.1 to 0.9 V_O		250		ns
t_f	Fall Time (*)	0.9 to 0.1 V_O		250		ns
t_{on}	Turn-on Delay (*)	0.5 V_i to 0.5 V_O		750		ns
t_{off}	Turn-off Delay (*)	0.5 V_i to 0.5 V_O		200		ns

(*) See fig. 1.



Digital I/O Specifications

Specifications

NI 653x (Continued)

Environment

Operating temperature	0 to 55 °C, DAQCard should not exceed 55 °C while in PCMCIA slot
Storage temperature	-20 to 70 °C
Relative humidity	10% to 90% noncondensing

Certifications and Compliances

CE Mark Compliance

NI 6527

These specifications are typical for 25 °C unless otherwise noted.

Digital Input

Optically isolated input channels	24, each with its own isolated ground reference
Maximum input voltage	28 VDC

Digital Logic Levels

Level	Minimum	Maximum
Input low voltage	0 VDC	1 V
Input high voltage	2 VDC	28 VDC

Input current	
5 V input	1.5 mA/channel max
24 V input	8 mA/channel max
Isolation	60 VDC channel-to-channel, and from computer

Digital Switch Output

Solid-state relay output channels	24, each with two terminals isolated from other channels
Relay type	Normally open form A solid-state relays
Maximum switching voltage	
AC	30 V_{rms} (42 V peak)
DC	60 VDC
Maximum switching capacity, 25 °C	120 mA
Common-mode isolation	60 VDC or 30 V_{rms} (42 V peak) channel-to-channel and channel-to-computer
On resistance	35 Ω maximum
Off leakage current (maximum)	200 nA
Relay set time (maximum)	3.0 ms
Relay reset time (maximum)	3.0 ms
Power-on state	Relays open by default, can be user-defined through software utility
Overcurrent protection on outputs	260 mA, typical

Power Requirement

+5 VDC (±5%)	500 mA, maximum
Power available at I/O connector	+4.5 to +5.25 VDC, fused at 1 A

Physical

Dimensions (not including connectors)	
PCI-6527	17.5 by 10.7 cm (6.9 by 4.2 in.)
PXI-6527	16 by 10 cm (6.3 by 3.9 in.)
I/O connector	100-pin keyed female

Environment

Operating temperature	0 to 50 °C
Storage temperature	-20 to 70 °C
Relative humidity	10% to 90%, noncondensing

Certifications and Compliances

CE Mark Compliance

NI 650x

These specifications are typical for 25 °C unless otherwise noted.

Digital I/O

Number of channels	
NI 6503	24
NI 6507, NI 6508	96
Compatibility	5 V TTL/CMOS
Power-on state	Input

Digital logic levels

Level	Minimum	Maximum
Input low voltage	-0.3 V	0.8 V
Input high voltage	2.2 V	5.3 V
Output low voltage (Iout = 2.5 mA)	–	0.4 V
Output high voltage (Iout = 2.5 mA)	3.7 V	–

Transfer rate

Bus	Maximum with NI-DAQ Software	Typical Sustainable Rate
PCI, PXI, DAQCard, ISA	50 kbytes/s	1-10 kbytes/s
DAQPad	250 bytes/s	175 bytes/s
Note: Transfer rate depends on the computer and software. The rates may vary due to programming language and code efficiency, CPU utilization, transfer methods, and so on. Please consult the user manual for specifics. The DAQPad-650x transfer rate is dependent upon available USB bandwidth.		

Handshaking	2-wire
Data transfers	Interrupts, programmed I/O

Bus interface

PCI, PXI, DAQCard, DAQPad, AT	Slave

Power Requirements

Device	+5 VDC (±5%)	Power Available at I/O Connector
6507/8 and PCI-6503	400 mA	+4.65 to +5.25 VDC, 1 A fused
DAQCard-DIO-24	15 mA	+4.65 to +5.25 VDC, 500 mA
PC-DIO-24	160 mA	+4.65 to +5.25 VDC, 1 A fused

Device	+9 to + 30 VDC	Power Available at I/O Connector
DAQPad-6507/8	150 mA at 12 VDC typical; 1 A max	+4.65 to +5.25 VDC, 1 A fused

Physical

Dimensions	
PCI-6503	12.2 by 9.5 cm (4.8 by 3.7 in.)
DAQCard-DIO-24	Type II PC Card
PC-DIO-24	11.7 by 10.6 cm (4.6 by 4.2 in.)
PCI-DIO-96	13.7 by 10.7 cm (5.4 by 4.2 in.)
PXI-6508	10 by 16 cm (3.9 by 6.3 in.)
PC-DIO-96	16.5 by 9.9 cm (6.3 by 3.9 in.)
DAQPad-6507/8	14.6 by 21.3 by 3.8 cm (5.8 by 8.4 by 1.5 in.)
I/O Connector	
NI 6503, except DAQCard	50-pin male
DAQCard-DIO-24	25-pin female PCMCIA
NI 6508, except PC-DIO-96	100-pin female 0.050 series D-type
PC-DIO-96	100-pin male ribbon cable

Environment

Operating temperature	0 to 55 °C, DAQCard should not exceed 55 °C while in PCMCIA slot
Storage temperature	-20 to 70 °C
Relative humidity	10% to 90% noncondensing

For information on static digital I/O in the VXI form factor, refer to the VXI Solutions Product Guide.

Certifications and Compliances

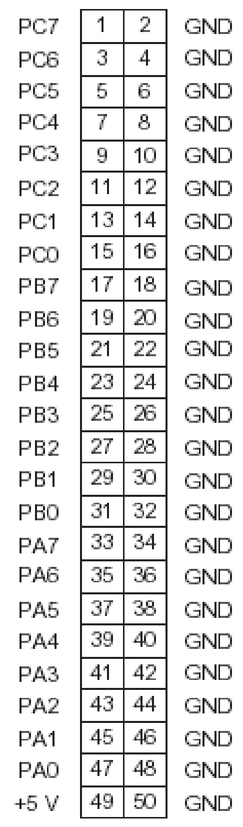
CE Mark Compliance CE

I/O Connector (PCI-6503)

The PCI-6503 has 50 pins that you can connect to 50-pin accessories with the NB1 cable.

PCI-6503 I/O Connector Pin Descriptions

Figure 3-4 shows the pin assignments for the PCI-6503 digital I/O connector using the NB1 ribbon cable.

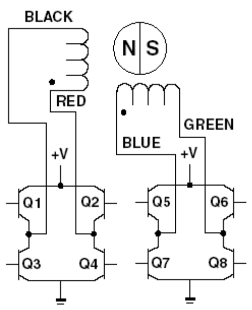


PC7	1	2	GND
PC6	3	4	GND
PC5	5	6	GND
PC4	7	8	GND
PC3	9	10	GND
PC2	11	12	GND
PC1	13	14	GND
PC0	15	16	GND
PB7	17	18	GND
PB6	19	20	GND
PB5	21	22	GND
PB4	23	24	GND
PB3	25	26	GND
PB2	27	28	GND
PB1	29	30	GND
PB0	31	32	GND
PA7	33	34	GND
PA6	35	36	GND
PA5	37	38	GND
PA4	39	40	GND
PA3	41	42	GND
PA2	43	44	GND
PA1	45	46	GND
PA0	47	48	GND
+5 V	49	50	GND

Figure 3-4. PCI-6503 I/O Connector Pin Assignments

Bipolar Winding

The two phase stepping sequence described utilizes a "bipolar coil winding." Each phase consists of a single winding. By reversing the current in the windings, electromagnetic polarity is reversed. The output stage of a typical two phase bipolar drive is further illustrated in the electrical schematic diagram and stepping sequence in figure 5. As illustrated, switching simply reverses the current flow through the winding thereby changing the polarity of that phase.



CW Rotation ↓

Bipolar Step	Q2-Q3	Q1-Q4	Q6-Q7	Q5-Q8
1	ON	OFF	ON	OFF
2	OFF	ON	ON	OFF
3	OFF	ON	OFF	ON
4	ON	OFF	OFF	ON
1	ON	OFF	ON	OFF

CCW Rotation ↑

Figure 5. Wiring diagram and step sequence for bipolar motor.

X.2 Control Program Source Code

The following is the main source code for the control application. Standard libraries and header files are not included. Some code was automatically generated by Microsoft Visual C++ 6.0

```
// MembranePrinter.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "MembranePrinter.h"
#include "MembranePrinterDlg.h"
#include "printercontroller.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
/////////////////////////////////////////////////////////////////////////////
// CMembranePrinterApp

BEGIN_MESSAGE_MAP(CMembranePrinterApp, CWinApp)
	//{{AFX_MSG_MAP(CMembranePrinterApp)
		// NOTE - the ClassWizard will add and remove mapping macros here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG
	ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMembranePrinterApp construction

CMembranePrinterApp::CMembranePrinterApp()
{
	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CMembranePrinterApp object

CMembranePrinterApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CMembranePrinterApp initialization

BOOL CMembranePrinterApp::InitInstance()
{
	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce the size
	//  of your final executable, you should remove from the following
	//  the specific initialization routines you do not need.

#ifdef _AFXDLL
	Enable3dControls();	// Call this when using MFC in a shared DLL
#else
	Enable3dControlsStatic();	// Call this when linking to MFC statically
#endif
//Create new printercontroller and Dialog window
	printercontroller pc1;
	CMembranePrinterDlg dlg;
	m_pMainWnd = &dlg;
	//Supply default input parameters to make typing easier
	dlg.m_Input1=1;
	dlg.m_Input2=200;
	dlg.m_Input3=0;
	dlg.m_Input4=30;
	dlg.m_Input5=1;

	//DoModal automatically updates the values in the keys corresponding to the
	//system parameters input by the user

	dlg.DoModal( );

	//now set the appropriate values in the printercontroller object
	pc1.setValues(dlg.m_Input1, dlg.m_Input2, dlg.m_Input4, dlg.m_Input5,
dlg.m_Input3);
	//run the system
	pc1.run();



	// Since the dialog has been closed, return FALSE so that we exit the
	//  application, rather than start the application's message pump.
	return FALSE;
}

// MembranePrinterDlg.cpp : implementation file
//

#include "stdafx.h"
#include "MembranePrinter.h"
#include "MembranePrinterDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();
```

```
// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMembranePrinterDlg dialog

CMembranePrinterDlg::CMembranePrinterDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMembranePrinterDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMembranePrinterDlg)
	//Original Data initilazation
	m_Input1 = 0;
	m_Input2 = 0;
	m_Input3 = 0;
	m_Input4 = 0;
	m_Input5 = 0;
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMembranePrinterDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CMembranePrinterDlg)
	DDX_Text(pDX, Input1, m_Input1);
	DDV_MinMaxInt(pDX, m_Input1, 0, 65000);
	DDX_Text(pDX, Input2, m_Input2);
	DDV_MinMaxInt(pDX, m_Input2, -6500, 6500);
	DDX_Text(pDX, Input3, m_Input3);
	DDV_MinMaxInt(pDX, m_Input3, 0, 65000);
	DDX_Text(pDX, Input4, m_Input4);
	DDV_MinMaxInt(pDX, m_Input4, 0, 65000);
	DDX_Text(pDX, Input5, m_Input5);
	DDV_MinMaxInt(pDX, m_Input5, 0, 65000);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMembranePrinterDlg, CDialog)
	//{{AFX_MSG_MAP(CMembranePrinterDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_DESTROY()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMembranePrinterDlg message handlers

BOOL CMembranePrinterDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here



	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CMembranePrinterDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

void CMembranePrinterDlg::OnDestroy()
{
	WinHelp(0L, HELP_QUIT);
	CDialog::OnDestroy();
}
```

```
// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CMembranePrinterDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM)
dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CMembranePrinterDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

// MembranePrinterDlg.h : header file
//

#if!defined(AFX_MEMBRANEPRINTERDLG_H__85C2FF2B_4BB5_4B8B_B39F_2
BA63298B443__INCLUDED_)
#define
AFX_MEMBRANEPRINTERDLG_H__85C2FF2B_4BB5_4B8B_B39F_2BA63298B
443__INCLUDED_
```

```
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CMembranePrinterDlg dialog

class CMembranePrinterDlg : public CDialog
{
// Construction
public:
	CMembranePrinterDlg(CWnd* pParent = NULL);	// standard constructor

// Dialog Data
	//{{AFX_DATA(CMembranePrinterDlg)

	//Tags defined for input going from top of dialog to bottom
//The tags allow the program to read in the user input.
	enum { IDD = IDD_MEMBRANEPRINTER_DIALOG };
	int		m_Input1;
	int		m_Input2;
	int		m_Input3;
	int		m_Input4;
	int		m_Input5;
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CMembranePrinterDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV
support
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CMembranePrinterDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnDestroy();
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	//}}AFX_MSG
```

```
        DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_MEMBRANEPRINTERDLG_H__85C2FF2B_4BB5_4B8B_B39F_2B
A63298B443__INCLUDED_)

// printercontroller.cpp: implementation of the printercontroller class.
// Written by Bill Pottle. Final Version- 7/11/2002
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"

#include "printercontroller.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif


/*
 * Includes:
 */

#include "nidaqex.h"
#include "time.h"
#include "windows.h"

//variables used. Note NIDAQ i16 data type used.
        i16 counter;
        i16 iStatus;
        i16 iRetVal;
        i16 iDevice;
        i16 iPort;
        i16 iPort2;
        i16 iPort3;
        i16 iMode;
        i16 iDir;
        i32 iPattern;
        i32 NumLoops;
```

```
          i16 iIgnoreWarning;
          i16 iYieldON;
          i16 numsteps;
          i16 stepdelay;
          i16 headdelay;
          i16 counter2;
          i16 ofsetdelay;


// Constructor for printcontroller class
printercontroller::printercontroller() {

           counter=0;
    iStatus = 0;
    iRetVal = 0;
    iDevice = 1;
    iPort = 0;
           //Port A
           iPort2 = 2;
           //Port C
           iPort3=1;
           //Port B- Not Used
    iMode = 0;
    iDir = 1;
    iPattern = 0;
    iIgnoreWarning = 0;
    iYieldON = 1;
//default values specified but not used
           numsteps=30;
           stepdelay= 1;
           headdelay= 200;
           counter2=0;
           ofsetdelay=0;

}
// function to set changable values

/*******************************************************************/
void printercontroller::setValues(i16 a, i16 b, i16 c, i16 d, i16 e) {
          NumLoops=a;
          headdelay=b;
          numsteps=c;
          stepdelay=d;
          ofsetdelay=e;
}
```

```
/********************************************************************/
//Delay Function

void delay (int dt)
{
//delays a specified number of milliseconds
        unsigned long final;
        final = dt+ GetTickCount();
                while (final>GetTickCount());
                return;

}

/********************************************************************/
//Motor up function
//Gives the stepper motor a specified number of steps to go up.
//Opposite of the MotorDown function
//This function is currently not used.


void MotorUp (int steps)
{
                counter=0;
                while ((counter < numsteps) && (iStatus == 0)) {
        //Note- Output goes on Pins C7, C6, C2 and C0

                // output - 1001 //132
                iPattern =132;
     iStatus = DIG_Out_Prt(iDevice, iPort2, iPattern);
     iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
      iIgnoreWarning);
                delay(stepdelay);

                // output - 0101 //68
                iPattern=68;
     iStatus = DIG_Out_Prt(iDevice, iPort2, iPattern);
     iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
      iIgnoreWarning);
                delay(stepdelay);

                // output - 0110 //65
                iPattern=65;
     iStatus = DIG_Out_Prt(iDevice, iPort2, iPattern);
     iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
      iIgnoreWarning);
                delay(stepdelay);
```

```
                  // output  - 1010 =  //129
                  iPattern=129;
        iStatus = DIG_Out_Prt(iDevice, iPort2, iPattern);
        iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
        iIgnoreWarning);
                  delay(stepdelay);
        iRetVal = NIDAQYield(iYieldON);

                  printf("Counter= %d", counter);
                  counter++;

   }



           iPattern=0;
        iStatus = DIG_Out_Prt(iDevice, iPort2, iPattern);
   // Output zero to turn off stepper motor to avoid overheating
                  iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
        iIgnoreWarning);

}


/******************************************************************/
//Motor down Function
//Gives the stepper motor a specified number of steps to go down.
//Opposite of the MotorUp function


void MotorDown (int steps)
{
                  counter=0;
                  while ((counter < numsteps) && (iStatus == 0)) {
           //Note- Output goes on Pins C7, C6, C2 and C0
                  // output  - 1010 =  //129
                  iPattern=129;
        iStatus = DIG_Out_Prt(iDevice, iPort2, iPattern);
        iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
        iIgnoreWarning);
                  delay(stepdelay);
        iRetVal = NIDAQYield(iYieldON);

                  // output - 0110 //65
                  iPattern=65;
        iStatus = DIG_Out_Prt(iDevice, iPort2, iPattern);
```

```
    iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
     iIgnoreWarning);
                delay(stepdelay);

                // output - 0101 //68
                iPattern=68;
    iStatus = DIG_Out_Prt(iDevice, iPort2, iPattern);
    iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
     iIgnoreWarning);
                delay(stepdelay);


                // output - 1001 //132
                iPattern =132;
    iStatus = DIG_Out_Prt(iDevice, iPort2, iPattern);
    iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
     iIgnoreWarning);
                delay(stepdelay);

                printf("Counter= %d", counter);
                counter++;

  }


       iPattern=0;
    iStatus = DIG_Out_Prt(iDevice, iPort2, iPattern);
  // Output zero to turn off stepper motor to avoid overheating
                iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
    iIgnoreWarning);



}



/*
 * Main function to run the system
 */
void printercontroller::run()
{

  /* Configure port A as output for printer head control, no handshaking. */

  iStatus = DIG_Prt_Config(iDevice, iPort, iMode, iDir);
```

```
   iRetVal = NIDAQErrorHandler(iStatus, "DIG_Prt_Config",
    iIgnoreWarning);

         /* Configure port C as output for stepper motor */
         iStatus = DIG_Prt_Config(iDevice, iPort2, iMode, iDir);

   iRetVal = NIDAQErrorHandler(iStatus, "DIG_Prt_Config",
    iIgnoreWarning);


         // Loop to set the print head moving back and forth.

   while ((counter2 < NumLoops) && (iStatus == 0)) {

                  // output 0 1 to move print head right
                  iPattern=16;
      iStatus = DIG_Out_Prt(iDevice, iPort, iPattern);
      iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
       iIgnoreWarning);
//Diagnostic Controlls not displayed in windows GUI mode
      printf(" The digital  chk pattern on port %d is set to (DECIMAL) %ld\n", iPort,
iPattern);
                  MotorDown(numsteps);
                  delay(headdelay);

                  // output 1 0 = 8 to move motor left. If ofsetdelay is 0, then do not move
the print head left at all.
                  //This is a default flag.

                  if (ofsetdelay==0) {
                           iPattern=0;
                  }
                  else {
                  iPattern=8;
                  }
      iStatus = DIG_Out_Prt(iDevice, iPort, iPattern);
      iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
       iIgnoreWarning);

      printf(" The digital pattern on port %d is set to (DECIMAL) %ld\n", iPort,
iPattern);


                  //MotorUp(numsteps);

                  //Delay coming back
```

```
        delay(headdelay+ ofsetdelay);

        iRetVal = NIDAQYield(iYieldON);
                        counter2++;

    }
                        // Write 0 to turn off port A and the print head motor.
                        iPattern=0;
        iStatus = DIG_Out_Prt(iDevice, iPort, iPattern);
        iRetVal = NIDAQErrorHandler(iStatus, "DIG_Out_Prt",
         iIgnoreWarning);
                         printf(" The digital pattern on port %d is set to (DECIMAL) %ld\n",
iPort, iPattern);

}

/* End of class */

// printercontroller.h: interface for the printercontroller class.
//
//////////////////////////////////////////////////////////////////////

#include "nidaqex.h"

#if
!defined(AFX_PRINTERCONTROLLER_H__CB11FFB3_3282_11D6_A450_00105
A0CA6B4__INCLUDED_)
#define
AFX_PRINTERCONTROLLER_H__CB11FFB3_3282_11D6_A450_00105A0CA6B4
__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class printercontroller
{
public:
            printercontroller();

void    run();
void setValues(i16, i16, i16, i16,i16);
};
```

```
#endif // !defined(AFX_PRINTERCONTROLLER_H__CB11FFB3_3282_11D6_A450_00105A0CA6B4__INCLUDED_)
```

X.3 Help File Source

The following is the rich text format file used in compiling the windows help file.

Table of Contents

Welcome to the Membrane Printer help file. Hopefully you don’t have to come here often, but you should find some simple tips that will help you in the operation of the device. For more information, refer to the published thesis or contact me at btp4@cornell.edu

***In general, the system should be run only with one loop with the default parameters. However, extra functionality has been added in case the system needs to take on other tasks at a later date.

***Also, if it is the first time the system is being used for the day, it is a good idea to lay down a ‘practice line’ on the paper holding the membrane. You can use deionized water or buffer solution to avoid having to reposition the syringe.

Common Problems

Here are some problems you might run in to:

Line stops after covering only half of the page:

This is probably due to either not having enough steps, or Windows needing to be restarted. If the fluid stops and there is a bubble at the end of the line, then either restart windows or increase headDelay. You should increase it in steps of 200 ms.

Fluid deposition is missing a couple of gaps:

This could be a problem with Windows, or with the syringe being positioned incorrectly (too far or too close to the membrane). If the problem is small, you can 'touch up' the fluid deposition by manually using the system. In this case, you will just have to turn the syringe plunger a quarter turn or so clockwise and eyeball the fluid line.

Print head not returning to the correct position (multi-looped operation):

Try adjusting delayOfset. A larger value will cause the print head to move more to the right; a smaller value will cause the print head to move more to the left. You should adjust it in progressively smaller steps starting from 200 ms and going down to ~5 ms.

Incorrect amount of fluid coming out:

Adjust the stepping constants. To get more fluid it's best to increase the number of steps, or decrease the delay between them. You can also try lowering the print head voltage, although this will affect the smoothness of your lines. To get less fluid do the opposite. The minimum value of step delay is 1 ms. The number of steps can be changed by 1 step at a time.

Stepper motor motion is choppy (you see dots on the paper instead of a line):

Try adjusting the step delay. Delays over about 4 ms will cause choppy motion. Other than that, perhaps try increasing the voltage. Increase the voltage in 1 volt increments.

Print head motion is choppy:

The print head voltage is probably too low. Voltages below about 7 V will cause choppy motion.

Print head motion too fast/slow:

Try adjusting the print head voltage. Higher voltages will result in faster movement.

Running the System

This is how you run the system under normal operation. First, you must make sure that all of the components are available and in place. Plug in the power supplies and make sure the variable one is set to the correct values.

1. Position the membrane into the device.

2. Position the syringe.

3. Set the print head all the way to the right and fill the syringe

4. Position the print head to the far right of the membrane, so that the needle is just touching the right edge.

5. Run the MembranePrinter.exe program.

6. Enter the parameters into the on-screen box. See the control algorithm for more information. The default parameters should usually suffice.

7. Click okay to start operation. Periodically read the value on the side of the syringe before and after operation to ensure the proper volume of fluid is being dispensed. This should be between 35 and 40 µL.

8. In rare cases you may have to ‘touch up’ the fluid deposition by manually bringing the print head over an unmarked area and slowly turning the syringe. This should be avoided if possible, but if necessary will still result in usable membranes.

9. Remove your membrane or reposition it to make another capture zone, and then manually move the print head back to the right. If possible, remove the membrane entirely before bringing the print head back to avoid splashing on the left side of the membrane.

10. After all membranes have been made clean the system.

11. Be sure to unplug both power supplies and turn off the variable supply, while leaving the dials set as they are.

Refilling the Syringe

Here is how you refill the syringe with the capture probe/streptavidin solution.

1. Put the immobilization solution in an eppendorf microcentrifuge tube. If desired, you can use the metal positioning plate. Place the tube in at the right side of the device and below where the needle will come in.
2. Manually set the print head to the right.
3. Raise the tube with the right hand so that the syringe needle contacts the fluid inside.
4. With the left hand, draw the fluid up into the syringe by raising the plunger. You do this by turning the motor/plunger attachment piece counterclockwise. Make sure you do not get any bubbles in the fluid. However, bubbles on top of the fluid are okay. It will usually be necessary to have a tight parafilm wrapping to secure the interface between the syringe barrel and the needle.
5. Lower the tube and take it away through the space in the right side of the device
6. Manually reposition the print head so that the needle is touching the right side of the paper.
7. Ensure that the screw connecting the motor shaft and the plunger is tight, and that there are no bubbles coming out of the needle tip.

Positioning the Syringe

The height of the syringe is very important to repeatable fluid deposition. After the membrane has been loaded into the system, do the following steps on the part of the scrap paper not covered by the membrane.

1. Insert the syringe so that the needle is just barely touching the paper. Set the lever on the right side that controls the print head height down. The needle should lightly scrape along the paper if you were to move it back and forth.
2. Tighten the attachment screw
3. Set the lever to the up position. Now freely slide the print head back and forth. The needle should not touch the paper or membrane.

Positioning the Membrane

The most important thing about putting the membrane in the device is that it is straight. Your fluid line will only be as straight as your membrane.

Fortunately, the regular printer page positioning may be used. Load the membrane from the front or the top and adjust the position with the dial on the left side. When putting the paper in under the grey roller, make sure the left and right sides go in at the same time. The paper should come out on top of the black rollers.

To position the membrane on a piece of scrap paper, first cut the membrane so that the dimensions are 20.5 cm by 7.5 cm per row of membranes you wish to print. Tape the membrane to a piece of letter size paper with the tape touching the right edge of the membrane and paper and the left edge of the membrane and paper touching each other. The laminated side should be down.

Cleaning the System

It is important to clean the system to avoid proteins getting stuck in the small needle. The needle can be removed and sonicated for 30 seconds. The syringe itself can be cleaned by flushing repeatedly with deionized water. This should be done after each use for the day, or when changing solutions.

Periodically the other mechanical components should be cleaned and oiled if necessary.

Important Reminders

There are several key points that must be followed or system performance can be greatly compromised. The following checklist should be observed each time before running the system and periodically between laying down lines of fluid.

-*Shut down all windows programs*- The timing function is run from software, so any other programs running on the system at the same time will cause non-uniform motor step delays which can lead to bad fluid deposition. This includes sharing files on the network. If the computer has a long uptime, it may be necessary to restart.

-*Ensure that the connector between the syringe plunger and motor screw is tight*- If the connector is loose, the right side of the fluid deposition will be spotty and may skip a space.

-*Make sure that the fluid is even with the syringe tip*- There should not be a bubble when you start the system. Similarly, if there is no fluid in the needle or bottom of syringe, there will be gaps in the right side of the fluid deposition.

-*Make sure all voltage dials are at their appropriate setting*- Small changes in voltage can cause significant differences in system operation.

-*Make sure that the syringe/needle interface is sealed*- A piece of parafilm wrap should be wound around the interface multiple times to create a good seal. Without a seal you risk some of your fluid being ejected from the sides. This will result in uneven deposition and wasted probes.

-*The amount of fluid ejected from the syringe does not always equal the amount deposited on the paper*- If the print head motion is fast enough to bring the print head all the way to the left while the motor is still stepping, then some fluid will be not deposited on the membrane but rather on motor casing on the far left side of the device.

-*Make sure that the external power supplies are unplugged when not in use*- The device should be fine under normal operation, but if the program exits abnormally or if for some reason a bad connection is made or voltage is accidentally increased, the system can overheat and components can burn out. The system could also be susceptible to power spikes or failures that can occur in electrical storms or power outages. The safest way to combat this is to only plug it in when it is being used.

Control Algorithm

Understanding how the membrane maker device is controlled is critical to fine tuning its operation. Basically, the computer steps through the following algorithm. The parameters highlighted in red are controllable by the user through the graphical user interface.

```
Do numLoops times {
        Set printer to move left (write 01 to port A)
        Do numSteps times {
                Step through (1001-0101-0110-1010) with a stepDelay each time
        }
        Delay headDelay
        If (delayOfset != 0) {
                Set printer to move right (write 10 to port A)
        }
        Delay headDelay+delayOfset
```

Note: You usually won't have to change many of the variables. The device supports looped operation, but you should probably use only one loop. The default parameters should work in most situations.

```
}
```

XI. References

1. ***Biochemistry*** **Stryer, L.** 4th Edition W.H. Freeman and Company, New York. 1999

2. **BEE 659: Biosensors Class Notes, spring 2002.** Baeumner, A. Biological and Environmental Engineering Department, Cornell University, Ithaca, NY.

3. **BD Biosciences Company Tissue Culture Surface Chemistry Product Brochure.** BD Biosciences, San Jose, CA.

4. **Molecular Recognition between Genetically Engineered Streptavidin and Surface-Bound Biotin.** Perez-Luna, Victor H. *Journal of the American Chemical Society* (1999), 121(27), 6469-6478.

5. **Nucleic acid sequence-based amplification**. Compton J Cangene Corporation, Mississauga, Ontario, Canada *Nature* (1991 Mar 7), 350(6313), 91-2.

6. **LINOMAT 5 Product Brochure.** CAMAG Company Berlin, Germany 2002

7**. Automatic TLC Sampler 4 (ATS4) Product Brochure.** CAMAG Company Berlin, Germany 2002

8. **J- Epson Stylus Color 860 Product Information Sheet** Seiko Epson Corporation, Long Beach, CA

9. **Characterization of an Inkjet Chemical Microdispenser for Combinatorial Library Synthesis. Lemmo, et. al.**

10. **Ink-jet printing for the fabrication of amperometric glucose biosensors** Newman, J. D.; Turner, A. P. F. *Anal. Chim. Acta* (1992), 262(1)

11. **Microfab Company Microjet Product Information.** Microfab Company, Plano, TX.

12. **Inkjet Printing for Materials and Devices** Calvert, P.; *Chem. Mater.;* **(Review); 2001**; *13*(10); 3299-3305.

13. **Parallel Production of Oligonucleotide Arrays Using Membranes and Reagent Jet Printing** Stimpson, et. Al *BioTechniques*, Vol. 25, no. 5, pp. 886-890, November, 1998

14. Kimura, J.; Kawana, Y.; Kuriyama, T. *Biosensors* **1988**, *4*, 41-52

15. **Personal Communication**, CAMAG USA.

16. **Unpublished Undergraduate Research.** Peterson, D. Biological and Environmental Engineering Department, Cornell University, Ithaca, NY.

17. **Hamilton Company Product Information**. Hamilton Company Reno, NV.

18. **Personal Commuinication** Brozowski, E. Hamilton product manager.

19. **Spreading and Imbibition of Liquid Droplets on Porous Surfaces Clarke, et. al** *Langmuir* **2002,** *18,* 2980-2984

20. **Development of a Biosensor to Rapidly Test for Organisms Resistant to Antibiotics**. Hoffman, E. Pottle, B. Reilly, J. 2002. Unpublished work. Department of Biological and Environmental Engineering, Cornell University, Ithaca, NY.

21. **Selective Electroless Nickel Plating on Polyelectrolyte Multilayer Platforms** Wang et. al *Langmuir* **2001,** *17,* 6610-6615