

# **DESIGN AND FABRICATION OF A PIEZOELECTRIC MICRO DISPENSING PRINTING SYSTEM**

By

Shishir Gaurav

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

Master of Science

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg

Copyright © 2019 by Shishir Gaurav

## Abstract

This thesis presents a detailed discussion on designing the system architecture to control the piezoelectric micro dispensing system from Microdrop Technologies equipped with a 70  $\mu\text{m}$  nozzle diameter MD-K-140 dispenser head via a host computer. The system architecture is designed to control the two main modules of the inkjet printing system. They are, (i) the motor positioning control electronics to control the movement of micro dispenser head to print the structure and (ii) The micro-dispenser control electronics to control the dispensing parameters required to dispense the ink drops. An interactive user interface is designed create a structure, parse it and transmit the command to interact with these modules.

The structures were printed on a glass substrate using micro dispensing system from Microdrop Technologies. The printing was done at room temperature. A UTDaGII conductive silver nano inks was used for printing structures. The glass substrates were thoroughly cleaned by washing subsequently with DI water, acetone and isopropanol and kept for few hours before printing.

Two types of thermal sintering process were used for this work. They were, (i) An oven sintering, and (ii) A 500 W 120 V tungsten halogen lamp sintering. The sintering results were compared using these methods. Oven sintering process was done at different temperatures and time. Incandescent lamp sintering was done varying the distance of the printed structures from the lamp and the time. It was observed that, the conductivity of the printed structure using an incandescent lamp sintering at 1cm from the lamp for 2 minutes yielded almost the same value as obtained using oven sintering at 250  $^{\circ}\text{C}$  for 15 minutes. Maximum conductivity of around  $2.64\text{E}+05 \text{ S/cm}$  was obtained using oven sintering at 250  $^{\circ}\text{C}$  for 30 min.

## **Acknowledgements**

I would like to express my sincere gratitude to my advisor Professor Dr. Cyrus Shafai for their continuous guidance and encouragement which made the completion of this thesis possible. I am grateful to him for sharing with me his approaches to problem solving, and large amounts of knowledge during our meetings. I consider myself lucky to have had the privilege of being his student. Thank you.

I would like to thank Dwayne Chrusch for teaching me practical skills. I appreciate his help in the hands-on aspects and in assembly and designing frames during my thesis work.

I appreciate the friendship of some of the MEMS Research Group including Elnaz Afsharipour, Naeem Riaz, Byoungyoul Park, Yu Zhou and many more. I thank Elnaz for providing me with significant amounts of cleanroom training and helping me inside cleanroom. Thank you.

Finally, I would like to thank all my friends for all their support, encouragement, and friendship along the way.

## Table of Contents

Abstract .....	ii
Acknowledgements .....	iii
List of Tables .....	vii
List of Figures .....	ix
<b>Chapter 1: Introduction .....</b>	<b>1</b>
1.1. Printing Technology .....	1
1.2. Thesis Outline .....	3
<b>Chapter 2: Overview of Inkjet Printing Techniques .....</b>	<b>5</b>
2.1 Introduction .....	5
2.2 Background of Inkjet Printing .....	5
2.3 Working Principles of Inkjet Print Head .....	7
2.3.1 Continuous Inkjet Printing System .....	8
2.3.1.1 Binary Charge Deflection Printing System .....	10
2.3.1.2 Multiple Deflection Printing System .....	11
2.3.1.3 Hertz and Microdrop Printing System .....	12
2.3.2 Drop-On-Demand Inkjet System .....	12
2.3.2.1 Thermal Inkjet Printing System .....	14
2.3.2.2 Piezo Inkjet Printing System .....	16
2.3.2.3 Acoustic Inkjet Printing System .....	18
2.3.2.4 Electrostatic Inkjet Printing System .....	18
<b>Chapter 3: Hardware Component of Microdrop Dispensing System .....</b>	<b>20</b>
3.1 Introduction .....	20

3.2 Microdrop Dispensing System.....	20
3.2.1 Micro Dispenser Head (MD-K-140).....	21
3.2.1.1 Assembly of the Micro Dispenser Head (MD-K-140) .....	23
3.2.2 Driver Electronics (MD-E-30xx-130) Controlling the Dispenser Head	24
3.2.2.1 External Control of the Driver Electronics (MD-K-30xx-130) .	30
3.3 Monitoring the Drop Formation.....	31
<b>Chapter 4: System Model: Setting of Hardware and Software for Printing System .....</b>	<b>33</b>
4.1 Introduction.....	33
4.2 Experiment Setup: Assembly of X, Y stage of the Printer .....	33
4.2.1 Description of the Components .....	34
4.2.2 Software for Motor Controller .....	39
4.2.2.1 Setting Configuration of the Stepper Motor .....	40
4.2.2.2 Axis Calibration .....	41
4.3 System Architecture.....	42
4.3.1 User Interface Layer .....	43
4.3.1.1 Backend Implementation of Program Design.....	43
4.3.1.2 File Upload Feature.....	51
4.3.2 Control Layer .....	52
4.3.2.1 Handling Error/Exception .....	57
<b>Chapter 5: Fabrication of Conductive Lines using Silver Nanoparticles and Sintering...</b>	<b>62</b>
5.1 Introduction.....	62
5.2 Printing of Silver Nano-particle Conductive Ink .....	63
5.2.1 Droplet Substrate Interaction .....	64
5.3 Sintering Process.....	68

5.3.1 Experiments using Thermal Sintering in the Oven.....	70
5.3.2 An Incandescent Lamp Sintering .....	80
5.3.2.1 Experiment Results .....	81
5.3.3 Sintering Summary .....	85
<b>Chapter 6: Change Dispensing Parameters to Study the Drop Formation.....</b>	<b>88</b>
6.1 Introduction.....	88
6.2 Dot-to-Dot Spacing.....	88
6.3 Experiment Results .....	90
6.4 Sintering Summary .....	101
<b>Chapter 7: Conclusion and Future Work.....</b>	<b>103</b>
7.1 Conclusion .....	103
7.2 Future Work .....	105
<b>References .....</b>	<b>106</b>
<b>Appendix.....</b>	<b>109</b>
A. Code to design a line.....	109
B. Code to design a filled rectangle.....	110
C. Code to design a filled circle .....	112
D. Setting up the connection in servlet Context while deploying the server.....	116
E. A servlet file to process the request and response to and from the user interface .	123
F. Design of user interface in java server page (JSP) .....	137

## List of Tables

5.1 Key important properties of the silver nanoparticles ink from the datasheet .....	64
5.2 Dispensing Parameters to dispense silver nanoparticles ink drop .....	66
5.3 Key performance of silver nanoparticles ink from the datasheet .....	73
5.4 Calculated conductivity value in S/cm at 200 °C for 30 min.....	73
5.5 Calculated conductivity value in S/cm at 200 °C for 60 min.....	75
5.6 Calculated conductivity value in S/cm at 250 °C for 15 min.....	77
5.7 Calculated conductivity value in S/cm at 250 °C for 30 min.....	78
5.8 Calculated conductivity value in S/cm using 500-Watt Bulb, 1 cm from bulb for 2 min ...	82
5.9 Sintering using oven method at different temperature and time, drop dispensed at 150V, 30 μs pulse width and 100 μm drop spacing .....	85
5.10 Sintering using incandescent lamp and varying the printed structure distance from the bulb at different drop spacing for same dispensing parameters.....	86
6.1 Comparison of the change in line width and the thickness at different dispensing parameters printed with 50μm drop spacing at 150 V and 30 μs.....	92
6.2 Calculated conductivity value in S/cm at 250 °C for 30 min for structure printed with 50μm drop spacing at 150 V and 30 μs.....	93
6.3 Calculated conductivity value in S/cm at 250 °C for 60 min for structure printed with 50μm drop spacing at 150 V and 30 μs.....	94
6.4 Comparison of the change in line width and the thickness of the printed structure at decreased pulse width and same drop spacing.....	97
6.5 Calculated conductivity value in S/cm at 250 °C for 30 min for structure printed with 50μm drop spacing at 150 V and 20 μs.....	97

6.6 Comparison of the change in line width and the thickness of the printed structure	
at increased voltage level keeping same pulse width and drop spacing. ....	99
6.7 Calculated conductivity value in S/cm at 250 °C for 30 min for structure	
printed with 50µm drop spacing at 170 V and 30 µs.....	100
6.8 Comparison of sintering result when drop dispensed at different dispensing parameters,	
50 µm drop spacing. Pattern sintered at 250 °C for 30 min using oven sintering method ..	101
6.9 Comparison of line thickness and line width at different dispensing parameters.....	101
6.10 Comparison of result when drop dispensed at 150V, 30 µs pulse width and two different	
drops spacing. Pattern sintered at 250 °C for 30 min using oven sintering method .....	102



## List of Figures

2.1 Classification of different inkjet-based printing technologies .....	8
2.2 Continuous inkjet printing: a multiple-deflection system.....	11
2.3 Drop-on-demand inkjet printing system .....	13
3.1 Diagram depicting the core of the micro drop generator.....	21
3.2 Drop generation process .....	22
3.3 Micro dispenser head MD-K-140 with a vertical holder MD-H-712-H.....	23
3.4 Front part of the dispenser electronics .....	24
3.5 Rear part of the dispenser electronics .....	25
3.6 Control keys of the touch screen to automatically fill and empty the dispenser head.....	28
3.7 Control keys of the touch screen to control the dispensing parameters.....	29
3.8 Assembly of monitoring system provided by the Microdrop Technologies.....	31
3.9 Connection for the USB camera type MD-O-539-USB .....	30
3.10 A water stream coming out from drop-on-demand micro dispensing system from Microdrop Technologies .....	32
4.1 Assembly of the frame .....	35
4.2 A frame with attached end stops connection .....	36
4.3 A motor controller RAMPS Board .....	38
4.4 System architecture. Host computer interact and control the controllers .....	42
4.5 A User interface Page designed in Java. ....	44
4.6 Option to select Line as a design pattern on user interface page. ....	46
4.7 Option to select rectangle as a design pattern on user interface page.....	48
4.8 A 3×3 rectangle pads with a 60 mm line created and sintered on a glass substrate .....	49
4.9 Option to select circle as a design pattern on user interface page .....	50

4.10 A filled circle of radius 5mm created and sintered on a glass substrate .....	51
4.11 A flowchart to understand mainly the interaction with motor controller .....	53
4.12 A flowchart to understand the interaction with the motor controller and dispenser controller .....	56
4.13 A flowchart to understand the interaction with the motor controller while handling error/exceptions.....	58
4.14 A text area and a text box on user interface to show the response and fire the command. ....	59
4.15 Emergency stop design implementation .....	61
5.1 A set of six parallel lines separated by 3mm printed on a glass substrate and sintered at 200 °C for 30 min .....	65
5.2 A micro dispensing system from Microdrop Technologies generating a 95 µm drop diameter of UTDAgIJ conductive silver nano ink at 150 V, pulse width 30 µs and 100Hz .....	67
5.3 A sintered silver conductive silver nano ink drop. The drop is dispensed at 150 V, pulse width 30 µs, viewed using optical microscopes .....	68
5.4 Four stages of silver nanoparticle structure during the oven sintering .....	70
5.5 Line sintered at 200 °C for 30 min viewed from optical microscope .....	71
5.6 Resistance (Ohm) vs the length of the wire (µm). Resistance measured for 100 µm drop spacing sintered at 200 °C for 30 min. ....	72
5.7 Resistance (Ohm) vs the length of the wire (µm). Resistance measured for 100 µm drop spacing sintered at 200 °C for 60 min. ....	74
5.8 Resistance (Ohm) vs the length of the wire (µm). Resistance measured for 100 µm drop spacing sintered at 250 °C for 15 min. ....	76
5.9 A 100 µm drop spacing line with thickness 0.294 µm, viewed in an optical microscope ..	76
5.10 Resistance (Ohm) vs the length of the wire (µm). Resistance measured for 100 µm drop spacing sintered at 250 °C for 30 min. ....	78
5.11 Temperature vs Conductivity curve for 30 min at 200 °C and 250 °C .....	79
5.12 Silver particles not sintered properly, at 150 C for 1 hr, viewed in optical microscope....	80

5.13 Schematic diagram describing the assembly of the printed structure on glass substrate undergoing halogen lamp sintering .....	83
5.14 Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for 100 $\mu\text{m}$ drop spacing sintered at 1 cm from bulb for 2 min. ....	84
6.1 A Line drawn with 50 $\mu\text{m}$ drop spacing sintered at 250 °C for 30 min, viewed in optical microscope. Clearly indicating increase in the thickness.....	89
6.2 Cross profile of printed silver track as viewed in Alpha step 500 surface profiler. The silver track was printed with 50 $\mu\text{m}$ drop spacing, sintered at 250 °C for 30 min .....	90
6.3 Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for 50 $\mu\text{m}$ drop spacing at 150 V and 30 $\mu\text{s}$ , sintered at 250 °C for 30 min. ....	91
6.4 Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for 50 $\mu\text{m}$ drop spacing at 150 V and 30 $\mu\text{s}$ , sintered at 250 °C for 60 min. ....	94
6.5 Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for 50 $\mu\text{m}$ drop spacing at 150 V and 20 $\mu\text{s}$ , sintered at 250 °C for 30 min. ....	96
6.6 Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for 50 $\mu\text{m}$ drop spacing at 170 V and 20 $\mu\text{s}$ , sintered at 250 °C for 30 min. ....	99

# Chapter 1: Introduction

## 1.1 Printing Technology

There are many research works that have been carried out in recent years on printed electronics, in materials processing [1]–[3], and in bio applications [4] using printing technology. The printing techniques are mainly divided into two main categories depending on the nature of printing material and type of substrate material. They are, (i) Contact printing and (ii) Non-contact printing techniques. Contact printing techniques are mainly used for high budget and highly precise printing. The accuracy ranges to few nanometers, but the process is typically complex. Therefore, generally, a non-contact printing method is adapted, with inkjet printing being heavily explored. Inkjet technology is a cost-effective way to design a planar or even three-dimensional structures on a substrate [5].

Inkjet printing is an advanced patterning technique to fabricate electronic devices. In this technique, a solution of material is deposited through a micron sized inkjet nozzle head. Key advantages offered by the inkjet printing are low fabrication cost, high-throughput mass-production, a non-contact method, variable droplet size, no physical mask and less environmental problems. These advantages can result in inexpensive high-performance printed electronics. This technique can be a much better choice than the conventional photolithography for fabricating the electronic devices [6][7] if suitable performance can be achieved. While the production speed is not the same as compared to that of traditional printing techniques (such as screen printing), its flexibility makes inkjet printing system more suitable for fabricating electronics devices. [8].

The inkjet printing techniques are categorized as a continuous process (in which ink drops are continuously dispensed) and a drop-on-demand process (in which the ink drops are generated

when they are needed to be deposited on the substrate). The drop on demand system doesn't require any hardware for recycling unused ink, to synchronize drop deflection, and the charging (providing a voltage to charge the ink drops with a predetermined charge to accurately control the ink drop trajectories, discussed more in detail in chapter 2) as required in continuous inkjet system. Hence, this technique is less complicated and there is less risk of ink contamination as compared to continuous system. This makes drop-on-demand inkjet printing best to fabricate electronics.

The most important component of the inkjet printing system is the colloidal solution/chemical solution. This solution is also known as ink. Printing challenges can include low throughput, low resolution, improper ejection of ink drops and slow speed etc.. To avoid these challenges, proper selection of ink properties such as the viscosity and the dispensing head features are the key parameters. The ink must be properly dispensed using dispensing system. Metallic nanoparticles such as silver and gold nanoparticles have gained more importance as a conductive ink these days. They provide better conductivity than conductive polymers and can easily be operated at room temperature. They have been reported in many research articles with/without surfactant materials [9]. UTDAGIJ Conductive Silver Nano inks is used as a conductive ink for this work.

Various deposition patterns can be achieved using computer-controlled software. The software creates the image and converts the image into instructions for the printer. The inkjet printing system is also known as "direct-write". There is no need for mask, and it is a cost-effective way to design the patterns. In addition, this process also restricts waste elimination and hence contamination is minimized [10]. The application using inkjet technology for micro-manufacturing includes solar cells, flex circuits as electronics devices, micro optical lenses and optical waveguide as optical devices, metal coating and material development [11].

## 1.2 Thesis Outline

This thesis starts with an overview of inkjet technology, its history, and various deposition methods. This is covered in chapter 2.

This research was carried out with a piezoelectric Autodrop dispensing system from Microdrop Technologies GmbH equipped with a 70  $\mu\text{m}$  nozzle diameter MD-K-140 dispenser. Chapter 3 gives a detailed discussion of the Microdrop Technologies dispenser head and controlling electronic hardware.

Chapter 4 deals with the components to control the main modules of the inkjet printing system. The electronics components used for this purpose are stepper controller, stepper motors and the stepper drivers. This chapter also presents the computer-controlled software to enable different patterns of ink deposition. A Java application program was designed to create a pattern and convert that into instruction that motor controller uses. The system architecture is mainly to parse, execute and transmit user command and to offer an interactive user interface to control the printing system via a host computer.

Pre- and post-inkjet processing are essential for the printing process and are discussed in Chapter 5. Pre-inkjet processing is carried out to improve the adhesion between the substrate and the printed structures. The post-inkjet processing improves the electrical conductivity of the printed structures and their robustness. It is essential to pre-process the substrate before using that for printing purposes. Post-processing is the ink sintering process. Two types of sintering processes are used. They are, (i) An oven sintering and (ii) An incandescent lamp sintering. This chapter also presents the sintering experiments carried out at different temperature and time to study their effect on ink conductivity.

Chapter 6 discusses the dispensing of the ink drops from the nozzle at different dispensing parameters. Dispensing parameters such as voltage and pulse width are varied. The level of the voltage determines the level of the pressure which determines the acceleration and thus, velocity of the ink drop. The pulse width results in expansion and contraction of piezo material and thus influencing the size of the dispensed drop. This chapter also includes the experiments carried out to study the variation of drop formation with changing dispensing parameters.

## **Chapter 2: Overview of Inkjet Printing Techniques**

### **2.1 Introduction**

This chapter presents a detailed discussion of different methods of inkjet printing. It starts with the introduction on background of the inkjet printing and later covers the discussion on two major types of inkjet printing. They are, (i) continuous inkjet printing and (ii) drop-on-demand inkjet printing (thermal and piezo). Various types of these two inkjet printing techniques are also described and discussed in detail.

### **2.2 Background of Inkjet Printing**

Inkjet printing has been the versatile non-contact material deposition technique used for commercial purposes at home, offices and industrial applications. The basic principle of inkjet printing involves the injection of the liquid materials or inks. The inks may be a solute dissolved or dispersed in solvent. It is transferred from an ink chamber to the nozzle. It comes out through the nozzle on a target substrate to form the ink patterns [12].

The foundation of modern inkjet printing technology is credited to English physicist Lord Rayleigh and the Belgian physicist Joseph Plateau. They studied the mechanism of the break-up of liquid streams and described mathematically. Plateau was the first to publish on this field in 1856. He derived the relationship between jet diameter to drop size in 1865. Rayleigh also published many research papers in 1878 and studied the instability of jets. But, the origin of inkjet printing goes back to 18th century when in 1749, A. Nollet studied the effect of static electricity on stream of liquid drops. Later, in 1833, Felix Savart discovered an acoustic energy to form the liquid drops. This is the technique followed in some modern inkjet printers [13][14][15].



William Thomson, later Lord Kelvin, a Scots-Irish mathematical physicist in 1858 invented the first inkjet-like recording device using electrostatic forces. It was called as “Siphon recorder”. This was used for automatic recording of telegraph messages. This device used the continuous stream of ink.

It was only after fifty years that in 1951, that Elmqvist of Siemens-Elema company (US Patent 2,566,433) made the first commercial working devices based on Rayleigh’s break-up principle. It was a voltage medical voltage recorder. An analog voltage from a sensor was used to deflect the ink drops (The Siphon recorder, UK Patent 2147/1867).

The more detailed continuous inkjet was introduced by Richard Sweet of the Stanford University. He made a high frequency oscillograph. He applied a pressure wave pattern to an orifice to break the ink stream into ink drops of uniform size and spacing. Later, in 1968, A. B. Dick company made the first commercial continuous inkjet system. They used it for character printing.

It was also possible in the 1970’s to produce the ink drops when they are required instead of delivering the ink drops in a continuous way. Zoltan of the Clevite company in 1972 (US Patent 3, 683, 212), designed the first piezo-based drop-on-demand inkjet printer. A hollow tube of piezoelectric material was used, and a voltage was applied to squeeze the ink in the chamber and form an ink drop to dispense from the nozzle. It was Naiman from the Sperry Rand Company who invented drop-on-demand thermal inkjet printing in the 1960s. The ink drops were generated by boiling ink at certain time instances, but the company did not explain this idea to use it for commercial purpose [13]. Later, in 1979, Ichiro Endo and Toshitami Hara of the Canon company re-invented the drop-on-demand printhead, called Bubblejet and used the growth and collapse of a vapour bubble in an ink chamber to form the ink drops. Hewlett-Packard also developed similar

inkjet printers during the same time and designed the first successful low-cost inkjet printer. The first successful commercialised desktop inkjet printer was ThinkJet Printer of HP in 1984 [16].

### **2.3 Working Principles of Inkjet Print Head**

The main purpose of the inkjet system as discussed, is to produce the ink drops. The ink deposition methods depend on the ink properties such as drop volume and drop velocity. The inkjet printing techniques are mainly divided into two main categories. They are, (i) A continuous process (in which ink drops are continuously jetted) and (ii) A drop-on-demand process (in which the ink drops are generated when they are required on the substrate). The working principles of these two major types of inkjet printers are discussed in detail below. Continuous and drop-on-demand printers are sub categorise into many different types as shown in fig. 2.1.

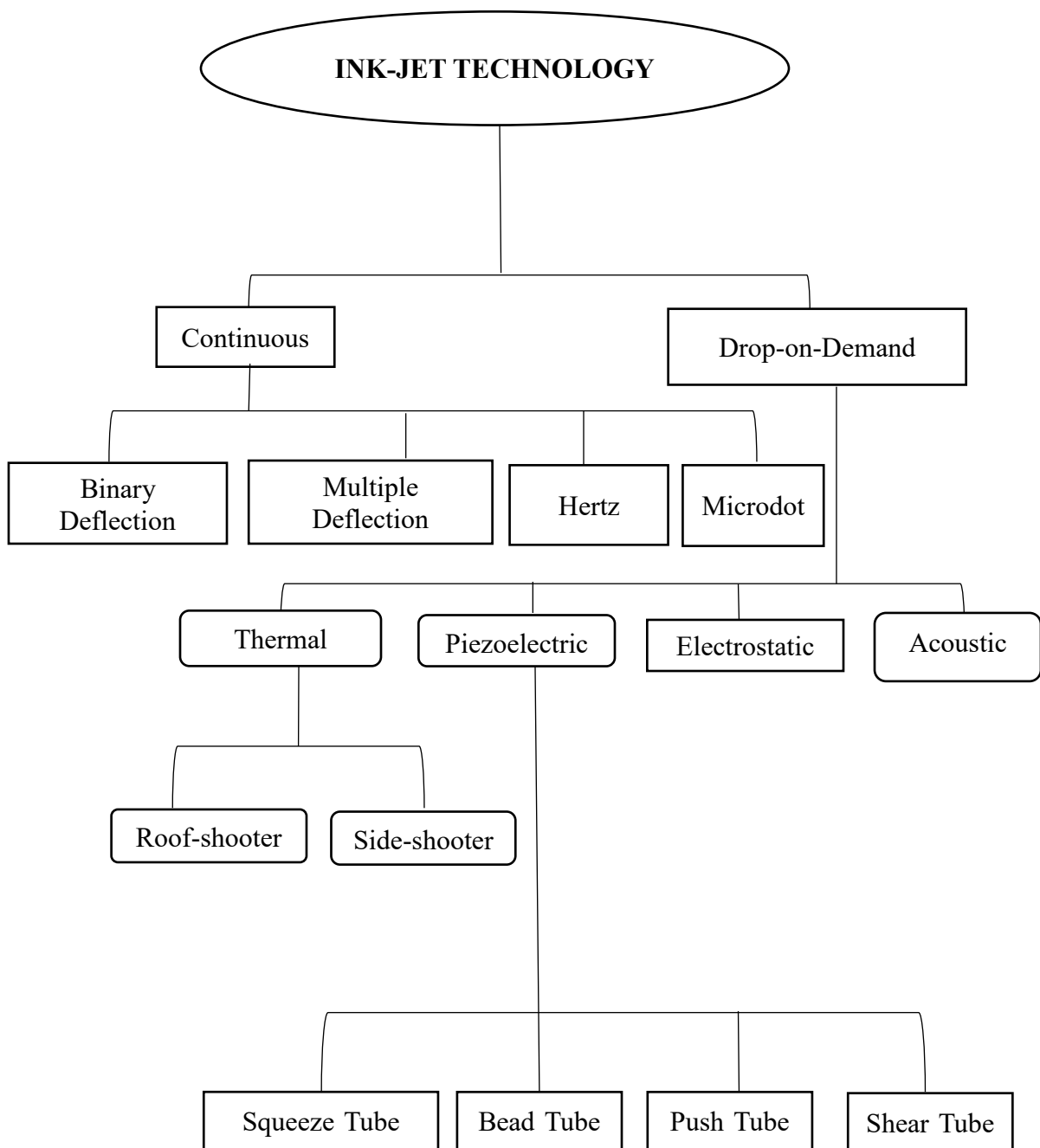


Fig 2.1: Classification of different inkjet-based printing technologies.

### 2.3.1 Continuous Inkjet Printing System

In the continuous inkjet printing process, the ink drops are constantly created and charged individually. An electric field is provided to separate the printing ink drops on the substrate and unused ink drops. Ink is transferred from the ink reservoir to the drop generator system. The ink is

dispensed from a small opening. This small opening is called as nozzle. It forms a continuous jet [17]. The ink stream ejecting from the nozzle is unstable and tends to break down into ink drops due to surface tension. This is due to instability phenomenon. This is due to their natural tendency to minimise the surface free-energy to reduce the surface area [18], [19].

An acoustic wave is generated by vibrating the piezoelectric crystal which results in forming the stream of ink drops. They are of uniform size and velocity. The ink drops are selected individually for printing. This is done using an induced electrical charge and electrostatic deflection. The ink drops are passed through the charging electrode and between the deflection electrodes. They gain a charge while passing through the charging electrode. The charge varies with the ink drops. After charging, the ink drops enter in between the deflection plates. The deflection plates are connected to a high voltage DC power source. This produces a constant electrostatic field. This electrostatic field creates a deflection of the charged ink drops from their original path. The amount of charge induced and the distance between the deflector plates determines that angle of deflection. The voltage applied to the deflection plates also determines the extend of deflection.

The speed of the ink drops, and the length of the deflector plates determine the exposure time to the electric field. The ink drops are deposited to the desired location onto the substrate. The deposition of ink forms the patterns. The unused ink drops are retrieved in a container. The unused ink drops can be recycled back into the main ink bottle for reuse [17].

The continuous inkjet system produces the ink drops which are twice the diameter of the orifice. The commercially available continuous system can produce the ink drops at a rate usually in the 80-100 kHz range, even as high as 1 MHz are also in use. The diameter of the ink drop is usually about 120  $\mu\text{m}$  which can also go as small as 20  $\mu\text{m}$ . This type of inkjet printing systems

are mainly used in textile printing, labeling product in food or medicine and beverage industry as the deposition speed is usually faster [20][21].

There are some disadvantages of the continuous printing system despite having such as good deposition speed. They are as follows:

- a) This system is not widely used in printed electronics as contamination of ink may occur due to the recycling process with exposure to environment.
- b) The continuous printing system is very expensive. A high maintenance cost is required for recycling process, to synchronize deflection and the charging and the dispensing system.

However, as discussed earlier, the continuous printing system performs printing at a high speed. Thus, this printing system is useful in industrial environment. The nozzle is not easily clogged, as the ink drops are generated continuously [16].

Continuous inkjet printing system are classified into different categories based on the method used for deflecting the ink drops. They are, (i) A binary charge deflection system, (ii) A multi charge deflection system, (iii) Hertz system and (iv) A microdot system. All these systems have a charging system, a deflection system, a dispenser system, a catcher system to recollect unused ink and an ink recirculation system. Domino, Imaje, VedioJet, and Toxot are some of the companies actively developing such printers. Recently, Nur Advanced Technologies demonstrated a billboard size inkjet printer using continuous inkjet printing system [16].

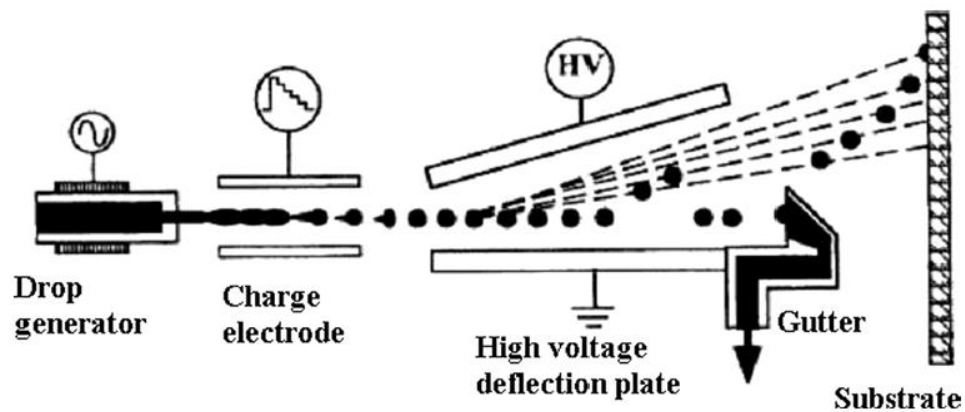
#### **2.3.1.1 Binary Charge Deflection Printing System**

In Binary charge deflection system, the drops are either charged or uncharged. The uncharged ink drops fall on the paper substrate and the charged ink drops are deflected into a container for recycling. In this process, a single nozzle can only print at one dot position. The different dot patterns on the substrate, are printed either by an array of jets, or by moving the

dispensing system relative to the substrate. This type of printer is slower as compared to the multi deflection system. This system mainly used for bar code printing and advertising purposes. Mead company first introduced it in 1973 [16]. Later, IBM developed and used this technology as a word processing output printer[22].

### 2.3.1.2 Multiple Deflection Printing System

In a multiple deflection system, the charged drops are deflected to the substrate at different levels. The unused ink drops are not charged. They fall directly into the container to be recirculated and reuse. In this system, only a single nozzle can print multiple dot positions. The deposition of ink drops forms the pattern. This system is faster than the binary deflection system [16]. They are used to print codes on canned foods [23]. Fig. 2.2 shows the multiple-deflection system.



Reproduced from [J. Mei, M. R. Lovell, and M. H. Mickle, "Formulation and processing of novel conductive solution inks in continuous inkjet printing of 3-D electric circuits," IEEE Trans. Electron. Packag. Manuf., vol. 28, no. 3, pp. 265–273, 2005].

Fig 2.2: Continuous inkjet printing: a multiple-deflection system [24]

### **2.3.1.3 Hertz and Microdot Printing System**

Professor Hertz of the Lund Institute of Technology in Sweden and his associates developed several continuous inkjet printing processes. They developed the process that had the ability to modulate the characteristics of the ink flow for gray-scale inkjet printing process. Hertz continuous inkjet printing system is the modified version of the binary deflection inkjet system. It is mainly used for color printing. The ink drops deposited are controlled through the volume of ink in each pixel. The color density is manipulated to achieve the desired gray tone [16] [19].

Lastly, in the microdot continuous inkjet system, the large and small diameter ink drops are dispensed from the nozzle, however, only the ink drops with small diameter entered the electrical field for selective ejection. The smaller drop diameter is produced without reducing the diameter of the nozzle. Hence, this system is for used as a selective ejection method [19].

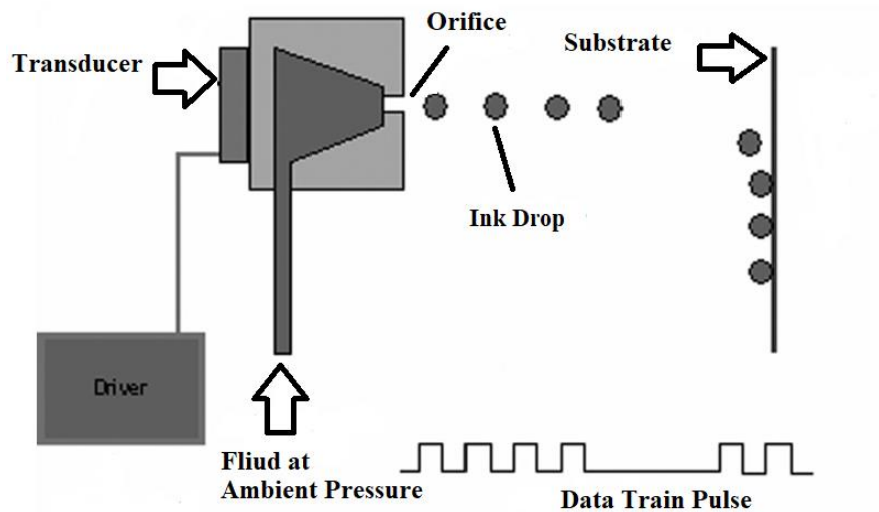
### **2.3.2 Drop-On-Demand Inkjet System**

There is another type of inkjet printing system which are commonly used. They are drop-on-demand inkjet printing system. These systems are used primarily in the office printer. Bubble Jets of Cannon, DeskJets of HP, and Stylus of Epson are the commercially available drop-on-demand inkjet printer systems. Hansell, in 1950 was the first who observed the production of the ink drops by means of an electromechanical process [16].

In this process, the ink drops are produced only when they are required. The deposition of ink forms a pattern on the substrate. This system doesn't require any hardware for recycling process, to synchronize deflection and the charging. Hence, it is less complicated and less risk of ink contamination, as compared to continuous system. In this process, the ink drops ejected from the nozzle is not continuous. The ink is transferred from an ink reservoir to the nozzle and is ejected through the nozzle only when a voltage pulse is applied to the transducer coupled with the nozzle

system. The transducer can either be a piezo or thermal [25]. This printing system is depicted in fig. 2.3.

The drop-on-demand inkjet system is classified into four major categories based on the mechanism used in forming ink drops. They are (i) Thermal, (ii) Piezoelectric, (iii) Electrostatic, and (iv) Acoustic systems. All these printers use a lower drop generation rate as compared to a continuous inkjet printing system. The mechanism of each of these printers are discussed in subsequent subsections. Thermal and piezoelectric inkjet printing systems are commonly used in market. The other two types of printers are hardly in use.



Reproduced from [J. Mei, M. R. Lovell, and M. H. Mickle, "Formulation and processing of novel conductive solution inks in continuous inkjet printing of 3-D electric circuits," IEEE Trans. Electron. Packag. Manuf., vol. 28, no. 3, pp. 265–273, 2005].

Fig 2.3: Drop-on-demand inkjet printing system [24]



### 2.3.2.1 Thermal Inkjet Printing system

The first type in this category is the thermal inkjet printer. In this inkjet printing system, the ink drops are generated similar to that of the process of water boiling to form water bubbles [19]. The transducer used in this process is a simple heating resistor. This heating resistor works as an electric heater. The temperature of the heater and the surrounding ink increases rapidly when current flows through it. The temperature increase is according as stated in Joule's effect. Temperature rises to about 350°C to 400 °C. This increase in temperature vaporizes the ink in the ink chamber. The vaporization creates a bubble at the surface of a resistor which causes an abrupt increase in pressure. The pressure pulse pushes the ink drops out from the nozzle. The vapour bubble disappears as the current passing through the heater is cut off. This led to decrease in the surrounding temperature. The ink drops are ejected from the nozzle and thus, generating a force to refill the ink again.

The mechanism of ink drop formation in the thermal inkjet system is a cyclic process. Here, in this process the nozzle is filled with ink. Tiny bubbles are left out in the nozzle. An electric heater is enclosed in the wall of the nozzle. When a voltage is applied to the heater, it produces heat. This heat makes the tiny bubble to inflate. The inflation of the bubble inside the nozzle is enough to force the ink drops to eject from the nozzle. The bubble explodes, and the ink drop ejects from the nozzle. The surface tension of the ink sucks more ink from the ink reservoir to refill the nozzle again. This procedure is fast. It takes less than 10 microseconds to complete [26][25].

The substrate used for fabricating thermal inkjet print head is patterned with various conductive and insulative layers. A resistive layer generating heat, made up of TaAl film, is sputtered on the substrate [20]. The heat is generated when an electric current pass through this resistive layer. This heat energy forms a vapour bubble in the ink. The ink drops are formed and

ejected from the nozzle. This heater layer is very fragile so many more layers are patterned on top of the heater. This is to protect it from thermal and mechanical damage. The damage may arise due to collapse of the bubble. Many research works have been carried out to use thermal inkjet printing systems to print mammalian cells, high-throughput cell patterning and the print of neural cell structures [27] [28]. There are a few advantages of thermal inkjet printing system. They are as follows:

- a) This system is suitable for smaller ink drop sizes and higher densities. This helps in achieving greater resolution.
- b) The cost of the print heads used in such systems is less expensive as compared to other drop-on-demand printers.

The ink used in thermal inkjet printing system must have a volatile material to form the vapor bubble. However, usually a water-based ink is used for such system which may contain dye as a colorant. Hence, thermal based despising system allows only limited range of the solvents in the ink.

The thermal inkjet printing system are divided into two categories depending on its configuration. They are, (i) A roof-shooter and (ii) A side-shooter. In the roof-shooter printing system, the orifice is located on top of the heater whereas, in the side-shooter printing system, the orifice is on a side located nearby the heater.

Hewlett-Packard, Lexmark, and Olivetti are the companies which makes printhead with the roof-shooter design. Canon and Xerox develop the side-shooter design [16].

### **2.3.2.2 Piezo Inkjet Printing system**

This is the most common type of inkjet printing system. In this method, a piezoelectric material is used as a transducer to generate the ink drop. These materials convert the electrical energy into mechanical deformation when a voltage is applied. This process is also called as inverse piezoelectric effect. This generates the pressure waves which propagates within the ink channel and produce the ink drops. The ink drops are ejected from the nozzle [29].

In this method, the ink drops are generated because of the contraction and expansion cycle of the piezo materials. These materials enclose the dispensing system. The size of the ink drops depends on the dispensing parameters such as voltage and the pulse width, and the characteristics of the dispenser head such as the nozzle diameter and the quality and the coupling of the piezo actuator. These dispensing parameters are adjusted to avoid the satellite drops and produce the good ink drops. The dispensing parameters are discussed as follows:

- a) The voltage level determines the level of the pressure which determines the acceleration of the liquid in the nozzle and the velocity of the ink drops. The control voltage required to dispense the ink drops depends on characteristics of the ink such as the viscosity and the surface tension and varies with the ink.
- b) The expanding of piezo materials creates a low-pressure pulse or a negative pressure in the channel. This process takes few microseconds. This low-pressure pulse propagates inside the channel and reaches the nozzle end. A part of this pulse is reflected from there.
- c) The contraction of piezo material creates an over-pressure pulse or a positive pressure that superimposes with the above low-pressure reflected pulse or wave. This interference overcomes the surface tension of the ink column and results in the formation of the ink drops [29] [30].

The piezo inkjet printing system is classified into four main types depending on the piezo material deformation mode. They are (i) A squeeze system, (ii) A bend system, (iii) A push system, and (iv) A shear system. The squeeze mode is the most basic configuration. In this configuration, the piezoceramic tube surrounds the nozzle and is radially polarized. In both the bend and push mode configuration, the directions of the electric field and piezoceramic plate deformation are in parallel. In bend mode, the diaphragm and the piezoceramic plates are bonded to form an array and an electromechanical transducer is used to generate the ink drops. In push mode, the piezoceramic plates expand and are pushed against the ink to generate the ink drops. A thin diaphragm is always there between piezo driver materials and ink to prevent any unwanted interactions between them. Dataproducts, Epson and Trident develops push mode type of printhead [16].

Lastly, in the shear mode configuration, the directions of the electric field and piezoceramic plate deformation are perpendicular to each other. The piezoceramic plate are deformed using shear action against ink and the ink drops are generated. The interaction between ink and piezo material is an important criterion in designing of such printhead [19].

Piezo inkjet printing system are mainly used in research and development works as it offers many advantages, as discussed above. Commercially available piezo inkjet printing systems are provided by companies such as MicroFab, Dimatix, and Microdrop Technologies (Norderstedt, Germany) [16] [29].

### **2.3.2.3 Acoustic Inkjet Printing System**

The second type of drop-on-demand inkjet printing system is an acoustic based thermal printing system. This is mainly used for adhesive coating. In this method, an acoustic energy is used to generate the ink drops. An acoustic lens is used to focus the ultrasonic beam on the surface of the fluid to produce the ink drops. In some of the acoustic printing system, spherical lens or Fresnel lens are also used to focus the beam. Fresnel lens structures are created by reactive ion etching of the silicon substrate. The acoustic printing system has some advantages. They are as follows:

- a) The nozzles in such system are safe from clogging.
- b) The ink drops size can be varied by changing the distance between the ink used and the transducer. This changes the focal spot diameter on the surface of the fluid. However, this printer is as commercially available [29][30].

### **2.3.2.4 Electrostatic Inkjet Printing System**

In this process, the electrostatic forces are used to generate the ink drops. This force is applied between an electrode and the nozzle. It attracts the free charges within the ink to its surface. The charged fluid is separated from the inkjet head as fine ink drops. Mostly, viscous inks are used in such printing system. The main advantage of this method is that the size of the ink drops is not controlled by the nozzle diameter. Hence, a high resolution can be obtained. The size of the ink drops can be smaller than the nozzle diameter. TTP and Seiko Epson mainly uses this type of inkjet printers for commercial uses [16][29][30].

Out of all these above inkjet printing systems, piezo inkjet printing system offers many advantages which are discussed as follows:

- a) There is no need to heat the ink to generate the ink drops as in the thermal inkjet system.  
This helps in preventing functional degradation of the solvent of the ink.
- b) Since the ink is not heated to a higher temperature, a broad range of inks can easily be dispensed using piezo inkjet printing system.
- c) The printhead of such systems has a longer life as it is not subjected to higher temperature.
- d) The maintenance cost is less, and they are commercially easily available.

## **Chapter 3: Hardware Component of Microdrop Dispensing System**

### **3.1 Introduction**

The dispensing system used for this research work was performed with a microdrop dispensing system from Microdrop Technologies (Norderstedt, Germany). This dispensing system was equipped with a piezo-based micropipette printhead with an inner diameter of 70  $\mu\text{m}$ . This chapter presents different hardware components of the dispensing system and their features.

### **3.2 Microdrop Dispensing System**

This section deals with the components and the functionality of the micro dispensing system from Microdrop Technologies (Norderstedt, Germany). It consists of a driver electronics MD-E-30xx-130 with pressure control, a dispenser head, a storage bin, a holder and a strobe diode with holder clip MD-O-505 and a CCD camera to monitor the ink drop formation. The micro dispenser head was delivered with storage bin mounted in the vertical holder MD-H-712. The dispenser head consists of a nozzle to dispense the ink drops. The assembly of this micro dispenser head are discussed later in detail in this chapter.

The driver electronics MD-E-30xx-130 triggers the micro dispenser head. The connection details of this driver electronics are discussed in section 3.2.2. This section also presents controlling the dispensing parameters such as voltage and pulse width. These parameters are controlled using the control keys of the touch screen of the driver electronics.

This driver electronics can be externally controlled using serial RS232 service interface. This feature is more discussed later in this chapter.

### 3.2.1 Micro Dispenser Head (MD-K-140)

This section deals with a simple schematic of the core of micro dispenser head. It is an ink drop generator. It is based on piezo driven inkjet printing technology. This dispenser head is equipped with a nozzle heater. A nozzle heater is a small heating coil. The heater is situated at around the last 10 mm of the droplet generator. The temperature detector is a thermo-resistance. Maximum temperature limit is 100°C. The increase in temperature reduces the liquid viscosity. A wide range of ink can be dispensed using this system. The ink viscosity in the range of 0.4 to 10000 mPas, volume ranging from 20 pl to 380 pl can be dispensed. Drop generation rate up to 6000 Hz can be achieved.

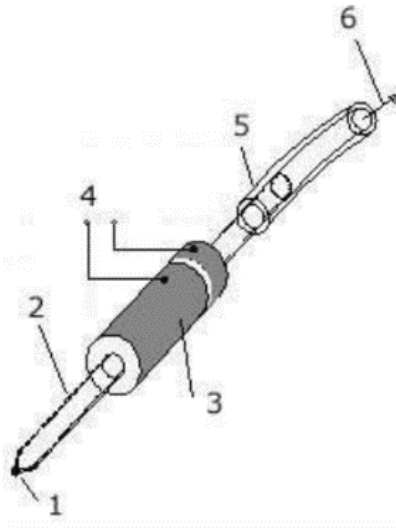


Fig. 3.1: Diagram depicting the core of the micro drop generator. 1: Nozzle. 2: Glass capillary. 3: Piezo actuator. 4: Supply voltage. 5: PTFE hose. 6: Connection to liquid reservoir [adapted from the device manual] [31].

The above figure 3.1 shows the schematic of the core of the micro dispenser head. The micro dispenser head consists of a glass capillary (2) surrounded by a piezo actuator (3). One end



of the glass capillary forms the nozzle (1) and the other end is connected to the liquid reservoir by a PTFE hose (5).

A supply voltage (4) with short electrical pulses are applied to the piezo actuator (3). The piezo actuator converts that into short pressure pulses. This pressure pulses, thereafter, expand from the glass capillary (2) through the liquid to the nozzle (1). The liquid is accelerated with a force generating the liquid jet ejecting from the nozzle (1). This ejection of the liquid jet decreases the pressure in the glass capillary (2), leading to deceleration. A small volume of liquid separates and forms the drop mainly due to the inertial force and surface tension.

The rest of the liquid is retracted into the nozzle (1) and oscillates there. The frictional loss leads the liquid to rest. The liquid is automatically refilled from the liquid chamber due to the capillary force into the nozzle (1).

Figure 3.2 shows the process in a sequence. The figure shows that the liquid first oscillates inward and thereafter it is accelerated with a force to generate a drop with tail. The tail becomes thinner during the process and breaks off [31].



Fig. 3.2: Drop generation process [adapted from the device manual] [31]

### 3.2.1.1 Assembly of The Micro Dispenser Head (MD-K-140)

Figure 3.3 shows the assembly of a micro dispenser head MD-K-140 with a vertical holder MD-H-712-H (4). The micro dispenser head consists of the nozzle (1). It is connected to a storage bin (5) using a PTFE hose (2). The hose transfers the liquid from the liquid reservoir (5) to the dispenser head (1). The liquid reservoir is tightly closed with a cover (6). The cover is made of PEEK thermoplastic. It is attached to the vertical holder (4). There are two openings in the cover (6). The first opening is tightly closed by a stopper (7). The second opening is provided with an air filter (8). The electrical connection (3) is connected to the dispenser head connection of the driver electronics (MD-E-30xx-130) [31]. The dispenser head connection of the driver electronics is shown in section 3.2.2. The connection of the driver electronics is discussed in next section.

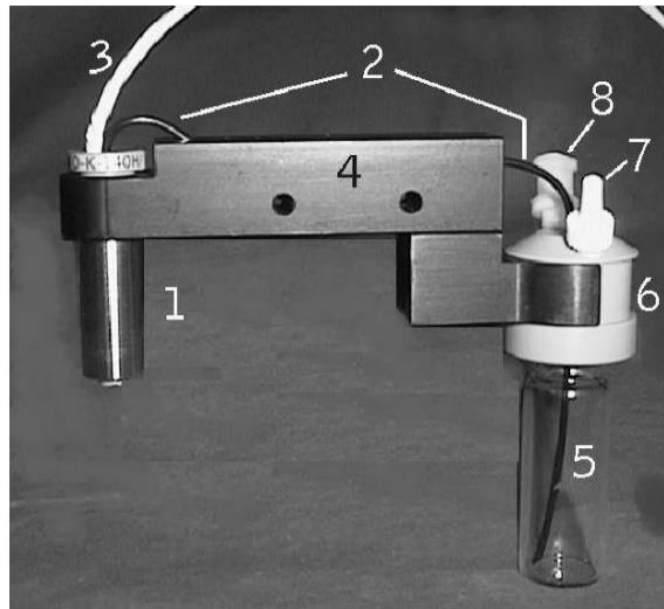


Fig. 3.3: Micro dispenser head MD-K-140 with a vertical holder MD-H-712-H [adapted from the device manual] [31]

### 3.2.2 Driver Electronics (MD-E-30xx-130) Controlling the Dispenser Head

This section discusses the driver electronics. This driver electronics trigger the dispenser head using the dispensing parameters such as voltage and pulse width. These dispensing parameters are required to dispense the ink drops from the nozzle. The nozzle is provided with the assembly of the micro dispenser head system. The assembly of micro dispenser head (MD-K-140) has already been discussed above. The driver electronics is interchangeably also called as micro dispenser controller in this work. The front screen of this driver electronics is a touch screen system. This is shown in fig. 3.4.



Fig. 3.4: Front part of the dispenser electronics: 1. Touch Screen 2. Start Menu

The rear part of the driver electronics is shown in figure 3.5. The main connection, as shown in fig. 3.5 (A), is for connecting the mains plug to the supply voltage. The nominal voltage required to operate this system is 100- 240 V.

The 6-pole plug at the rear side of the driver electronics, as shown in fig. 3.5 (B), is the dispenser head connection. This is connected to the nozzle, as shown in fig. 3.3 (3) to trigger it.

The nozzle is triggered using dispensing parameters such as voltage and pulse width. The nozzle is filled with the ink to dispense. These parameters can be changed using the control keys of the touch screen, as shown in fig. 3.7. A single rectangular pulse with determined rise time and falls time is used to trigger. The amplitude (voltage) and the length (pulse width) of the pulse are changed to form the ink drops. These parameters have already been discussed in detail in section last chapter 2 while discussing piezo inkjet printing system.

The 2-pole plug, as shown in fig. 3.5 (C), is the connector for the strobe diode assembly. This assembly is provided with a USB camera of type MD-O-539-USB to monitor the drop formation. The strobe diode assembly is discussed in detail in section 3.3.



Fig. 3.5: Rear part of the dispenser electronics. A: Main connection. B: A 6-pole plug for dispenser head connection for head 1. C: A two-pole plug for strobe output for USB Camera connection. D: Serial RS232 service interface. E: Air pressure connection for head 1 and head 2. F: A 6-pole plug for dispenser head connection for head 1. G: External trigger input. H: Ready output.

This driver electronics is also provided with the air pressure control. The air pressure control is used to automatically fill and empty the nozzle to and from the liquid reservoir. The air

pressure connection is shown in fig. 3.5 (E). The air pressure hose coming from the liquid reservoir as shown in above fig. 3.3 (8) connects the pressure control as shown in fig. 3.5 (E). The air pressure hose as shown in above fig. 3.3 (8) must be connected properly and tightly. The total pressure must be airtight.

The liquid reservoir initially contains the ink. The control keys of the pressure control, as shown in fig. 3.6, is used to transfer the ink from the liquid reservoir to the nozzle. Here, “Fill”, as shown in above fig. 3.6, means the overpressure to fill the nozzle from the liquid reservoir and “Empty”, as shown in above fig. 3.6, means the low pressure to empty the nozzle to the liquid reservoir. The pressure control generates a positive pressure to fill the nozzle and a negative pressure to empty the nozzle.

The driver voltage for the micro dispenser head is switched on and off using the control key “ON/OFF” key, as shown in fig. 3.7 (A). This “ON/OFF” key will start and stop the nozzle to dispense the ink drop. The ink is dispensed based on the dispensing parameters such as voltage and pulse width. These parameters can be adjusted using “Driver Setting” key, as shown in fig. 3.6 (D) in the toolbar of the main menu.

The driver electronics (MD-E-30xx-130) is externally controlled using a serial RS232 service interface, as shown in fig. 3.5 (D) to switch on and off the driver voltage for the micro dispenser head. This is discussed more in detail later in this chapter.

This driver electronics can trigger the two nozzle systems. Both nozzles can be used to dispense the ink drops. “Driver 1” and “Driver 2” as shown in fig. 3.6 and fig. 3.7, indicates the same. However, only one nozzle is used for this work. Fig. 3.6 (D) also shows some toolbars contain some more features. This is described as follows:

- Home: This key is for jumping to the main menu. The main menu is shown in fig. 3.7.

- Fill/Empty: This key is for the submenu (fill/empty), as shown in fig. 3.6.
- Backlight: This key is for providing the on and off feature of the backlight of the strobe diode. The backlight of the strobe diode is kept on during monitoring the ink drop formation.
- Driver Settings: This key is for jumping to the driver settings menu to adjust the dispensing parameters such as voltage and pulse width.
- Global Settings: This key transfers to the menu which contains features for adjusting the brightness of strobe diode and brightness of backlight. This menu also contains the features for controlling the nozzle heaters. The features of the nozzle heater are beyond the scope of this thesis. The nozzle temperature is kept at room temperature and remains constant for this work.
- Standby: This key jump back to the start screen.

The adjustable range of the dispensing parameters using this driver electronics are discussed as follows:

- Voltage: The range of the driver voltage is from -250V to +250V. The high voltage value results in high pressure generated by the piezo actuator.
- Pulse width: The range for the pulse width is between 1 and 200 $\mu$ s. The pulse width determines the duration of the piezo contraction and expansion cycle.
- Set Temp.(°C): This display value is the set value of the nozzle temperature in °C. The nozzle temperature is adjustable between 23°C and 100°C.
- Is Temp.(°C): This value shows the actual value of the nozzle temperature in °C.
- Frequency (Hz): This value shows the droplet rate or dispensing frequency. The adjustable range is between 1 ink drop per second and 6000 ink drops per second.

- Drops: The number of drops can be adjusted in the range of 1 to 10000.

The value of these dispensing parameters varies with the inks used for dispensing. The value adjusted for this thesis is discussed in chapter 4.

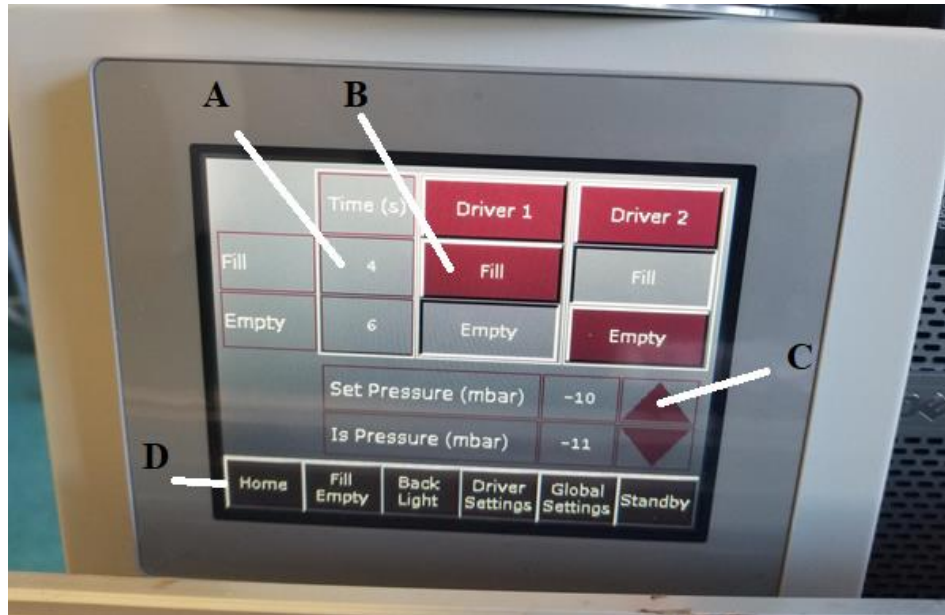


Fig. 3.6: Control keys of the touch screen to automatically fill and empty the dispenser head.

A: setting of the fill time. B: fill the nozzle. C: setting of the holding pressure. D: Tool bar of the main menu.



Fig. 3.7: Control keys of the touch screen to control the dispensing parameters. A: Control key to switch on and off the driver voltage. B: Control key to switch on and off the nozzle heater. C: Control key to select the operating mode (continuous, burst or gate (only for external trigger)). D: Control key to select triggering setting (internal or external).

The piezo actuator of the dispenser head can be either internal triggered or triggered using an external device over the external trigger input, as shown in fig. 3.5 (G). The ready output connection, as shown in fig. 3.5 (H), is for issuing a signal when the drive electronics are ready for the next input command. This mode of operation is beyond the scope of this work.

The piezo actuator of the dispenser head is internally triggered for this work. The “Intern” key as shown in fig. 3.7 (D) represents the same. There are two operating modes for the dispenser head at internal triggering. They are as follows:

- Continuous mode: The connected dispenser head dispenses at a constant frequency. The frequency is adjusted using the key “Frequency (Hz)”, as shown in fig. 3.7. The continuous mode is represented by “Cont.” key in the same figure.



- Burst mode: The connected dispenser head dispenses the pre-set number of drops after each release of a trigger pulse using the dispensing set parameters. The operating mode is set to burst for this work to carry out experiments. The “Cont.” key is clicked and the operating mode switches to burst mode and the button “burst” changes to grey.

### **3.2.2.1 External Control of the Driver Electronics (MD-E-30xx-130)**

The serial RS232 service interface connection as shown in fig. 3.5 (D) is for external control of the driver electronics. A one-Port FTDI USB to RS232 Serial Null Adapter cable from is used for this purpose. One end of this cable is connected to the service interface of the driver electronics, as shown in fig. 3.5 (D) and the other end of this cable goes to the host computer. This host computer runs the software to externally control the driver electronics.

The communication between the host computer and the driver electronics is established via a serial port communication. The communication preferences to set up the communication is discussed in detail in chapter 4. After establishing the communication, the driver voltage for the micro dispenser head is switch on and off to start and stop the nozzle to dispense the ink drop. Microdrop Technologies (Norderstedt, Germany) provided the command list for this purpose. The command is sent via a serial port communication to execute this task. The interaction with the driver electronics using command is discussed in detail in next chapter 4.

The main screen display of the driver electronics is not active during this process. The dispensing parameters such as voltage, pulse width and number of drops are pre-set before establishing the communication using control keys of the touch screen, as discussed above. The nozzle head is now started and closed using command to dispense the ink drop.

### 3.3 Monitoring the Drop Formation

A strobe diode and a USB camera of the type MD-O-539-USB is used to monitor the ink drop formation. The ink drops are dispensed from the nozzle. XIMEA Windows Software Package is installed on the host computer and xiViewer used for monitoring the ink drops.

Figure 3.8 shows the assembly of the monitoring system and fig. 3.9 shows the connection of the camera. After connecting the camera with the host computer, the image can be viewed using xiViewer. The image viewed using xiViewer is shown in fig. 3.10. An LED assembly is in the holder clip of the strobe diode. A short flash is generated in synchronous to the driver electronics and an optically frozen drop is visualized from the camera.

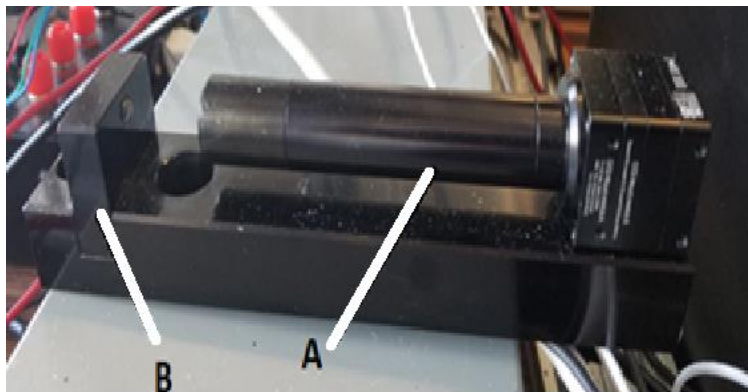


Fig. 3.8: Assembly of monitoring system provided by the Microdrop Technologies. A: CCD camera (MD-O-539-USB), B: Strobe diode with holder clip (MD-O-505)



Fig. 3.9: A: Connection for the USB camera type MD-O-539-USB. One end of the cable is connected to the camera whereas another end is with the host computer running the xiViewer window.

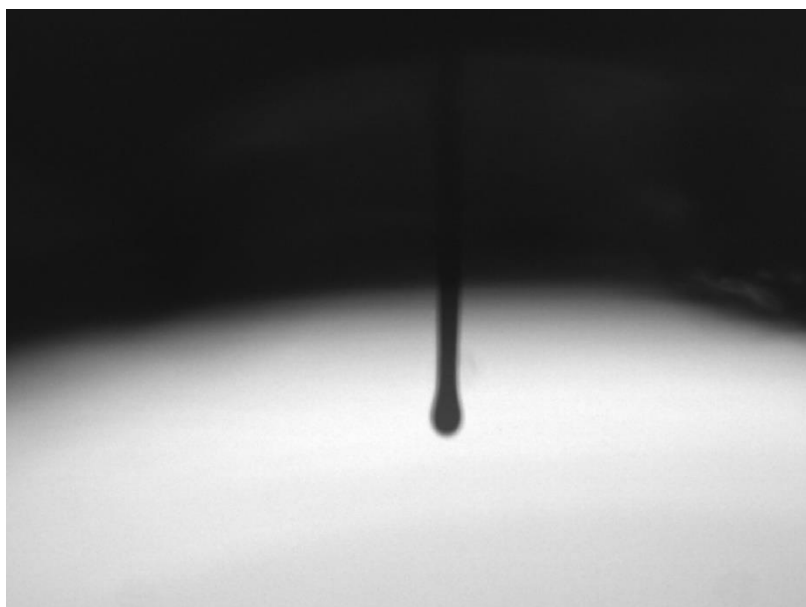


Fig. 3.10: A water stream coming out from drop-on-demand micro dispensing system from Microdrop Technologies. This picture is captured using USB camera of the type MD-O-539-USB and viewed on a host computer running xiViewer. The water is dispensed from the nozzle at 50 V, 25  $\mu$ s pulse width and at 100 Hz.

## **Chapter 4: System Model: Setting of Hardware and Software for Printing System**

### **4.1 Introduction**

The background and working principle of different types of inkjet printing systems is discussed in the previous chapters. The discussion also included the hardware components of the micro dispensing system from Microdrop Technologies (Norderstedt, Germany). This system was used to dispenser the ink drops for this work.

This chapter deals with assembly of different components of the printing system. The discussion is mainly concentrated on the assembly of the two main modules of an inkjet printing systems. They are, (i) Motor Positioning Control Electronics and (ii) Microdrop Dispensing System. This chapter includes the assembly of the X, Y stage for printing system. Section 4.2 deals with this experiment setup. It also includes the discussion on the components used in these modules.

The mechanism to control these modules has also been discussed in this chapter. A system architecture is designed to communicate and control these modules. The detailed discussion is carried out on the design process in this chapter.

### **4.2 Experiment Setup: Assembly of X, Y stage of the Printer**

The section deals with the assembly and design of the X, Y stage for the system. This has been done to perform a good and an accurate printing process. The assembly of the micro dispenser head has already been discussed in last chapter. The same assembly of the micro dispenser head is incorporated to the X, Y stage. The different components used for the process includes X-axis, Y-

axis and Z-carriage, substrate, stepper motors to control the movement of the carriage, electronics parts, stepper motor controllers and end-stops. The below section deals with these.

#### **4.2.1 Description of The Components**

##### **(a) Frame:**

The frame provides the stiffness to the printer. The three axes and the assembly of the micro dispenser head are added to frame. Figure 4.1 shows the complete assembly of the frame. The frame consists of 8 mm-metric-acme-lead-screw combined with printed parts. The three stepper motors are attached to the three X, Y and Z axis.

The X-axis traverses the attached nozzle to the left and right within the frame. The X\_MIN is the left side movement of X axis and X\_MAX is the right-side movement of the X axis. For example, X frame moves 10 mm to the left side from the current position for X -10 movement and X frame moves 10 mm to the right side from the current position for X +10 movement. The range of X axis movement is from -70 mm to 130 mm from the working zero coordinates. The direction of the axis is shown in below fig. 4.1.

The Y-axis traverses the attached Z axis, as shown in fig. 4.1 to back and forth within the Y-axis frame. Y\_MAX and Y\_MIN direction is shown in fig. 4.1. Y\_MAX is the moving away from the nozzle assembly and Y\_MIN is the movement towards the nozzle assembly. The nozzle assembly is shown in fig. 4.1(E). For example, Y axis moves 10 mm towards the nozzle assembly for Y -10 movement and Y axis moves 10 mm away from the nozzle assembly for Y +10 movement. The range of Y axis movement is from -20 mm to 250 mm from the working zero coordinates. The lamp sintering assembly is also installed on the Y axis. This assembly is discussed in detail in chapter 5.

A substrate is installed on the Z axis which moves up and down. The maximum value of Z axis movement is 80mm upwards from the base position. The Z axis is kept at base position in the below fig. 4.1. The Z axis cannot move below the base position. Z positive moves upwards, whereas Z negative moves downwards. However, the Z axis is always kept constant at a 1 mm distance from the nozzle to carry out the experiments for this research work. The result of the experiments is described in chapter 4 and chapter 5.

The mechanical and the optical end stops are attached the X and Y axis as shown in fig. 4.2. The end stops connection is discussed in detail in latter part of this section.

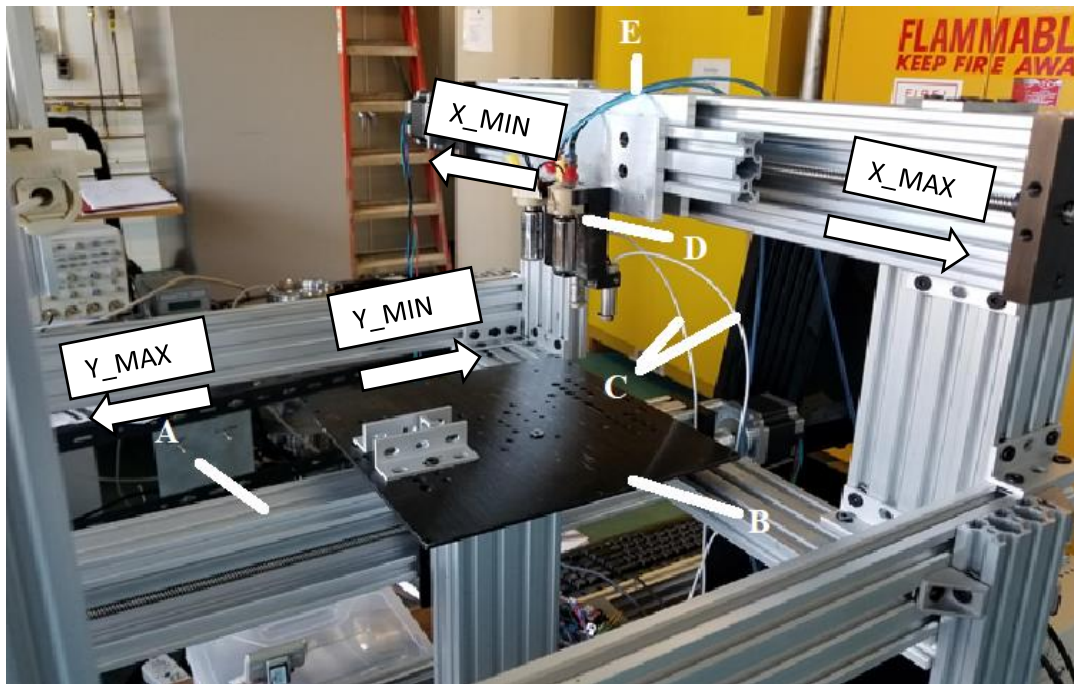


Fig. 4.1: Assembly of the frame. A: The Y axis traversing the black color substrate holder attached to the Z axis back and forth (Y\_MAX and Y\_MIN direction as shown), B: The substrate holder attached to the Z axis, C: Dispenser head connection, D: The nozzle assembly. E: The air pressure hose connected for air pressure control of the dispenser head.

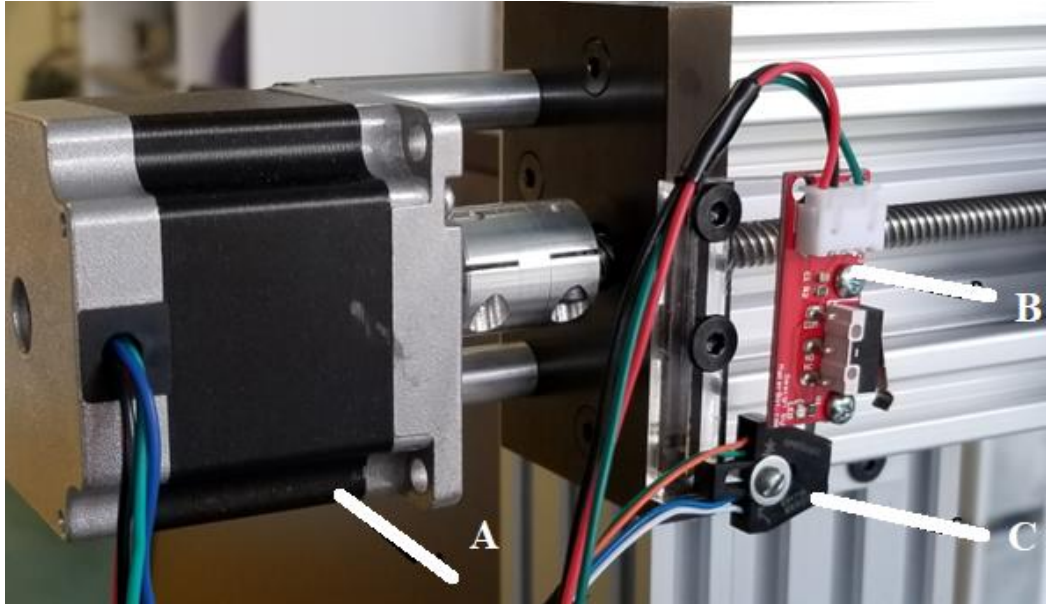


Fig. 4.2: Frame with attached end stops connection. A: Nema 23HM22-2804S stepper motor. B: Commercially available Mechanical end stop for emergency situation. C: Optical end stop OPB70AWZ.

#### (b) Control Electronics for positioning system

Electronics to control the positioning system of the printer can be categorized into four different areas. They are, (i) Stepper controller, (ii) Stepper motors, (iii) Stepper drivers, and (iv) end stops. These control electronics are designed to position the dispenser head and control and regulates the three axis stepper motors with good speed and good accuracy. The components of the control electronics are discussed below:

##### **Stepper motor:**

The three Nema 23HM22-2804S stepper motors are used to control the three axis, X, Y and Z. These axes are installed on the frame, as shown in fig. 4.1. The installation of the stepper

motor is shown in fig. 4.2. Here, black and green wires form one pair and red and blue wires form another pair of the stepper motor.

### **Stepper motor controller:**

The stepper motor controller is the main part of the operation. An Arduino microcontroller with an add-on board is used as a motor controller for this work. The add-on board is a RepRap Arduino MEGA 2560 Pololu Shield or RAMPS for short. It is an 8-bit microcontroller circuit board to translate the computer commands into movement actions. The firmware is a program which is installed on this controller to achieve the movement actions. The detailed discussion on the firmware program is done in later part of this chapter.

Figure 4.3 shows the RAMPS board. The add on components such as stepper drivers and end-stops are attached on it. All the three stepper motors of the three axes are installed on the board at the X, Y and Z axis shown in fig. 4.3 (B, C, D). The red and blue wires which form one pair, are connected to the 2A and 2B section on the board and black and green wires which for another pair, are connected to the 1B and 1A section on the board.

The Ramps board is the most commonly used controller for cartesian movements. It is plugged in to the Arduino MEGA 2560 and is connected to the host computer using a USB-to-serial converter. The controller board is powered using a 12 V power supply. This board is very cheap as compared to other commercially available boards, while most of the other boards have built-in drivers, this stepper motor controller has the slot for add on drivers. Thus, stepper motor drivers can be played around with many numbers of times and with many other drivers.



## Stepper drivers:

The next component is the stepper drivers. These drivers are installed on the controller board. These drivers are chips which acts as a buffer between a stepper motor and the controller. These chips send the simplify signal to the stepper motor to move. Each stepper driver is connected to a stepper motor. The three jumpers are installed under each stepper driver to control the accuracy of the stepper motor. The stepper motors are controlled with the controller. Three Pololu A4988 stepper motor drivers are used for this work. These are installed on the controller as shown in fig. 4.3.

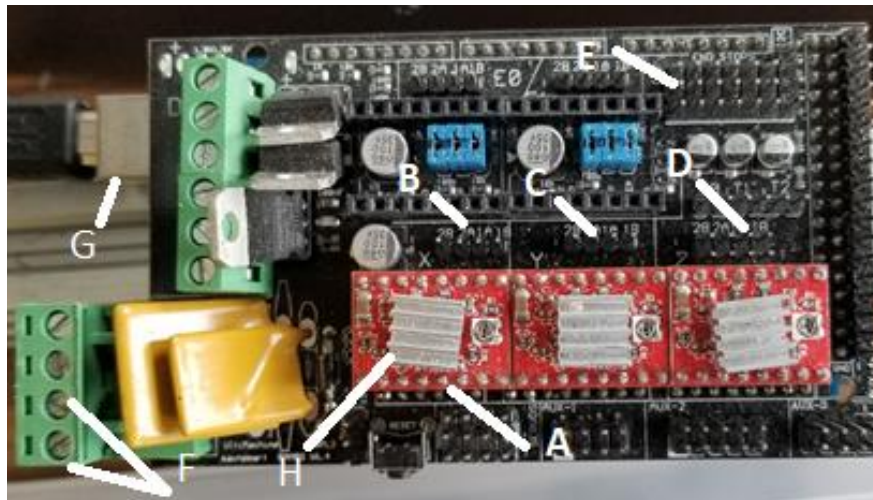


Fig. 4.3: A motor controller RAMPS board. A: Three Pololu A4988 stepper motor drivers for each of the X, Y and Z axis. B: X axis motor slot. C: Y axis motor slot. D: Z axis motor slot. E: Six End stops slot, two each for X, Y and Z axis. F: Power supply connector. G: USB connector.

## End Stop:

Another piece of the hardware is the end stops. These are very small and simple circuit boards with a switch on it. This tells the motor controller to stop when the motor moves too far.

This prevents the motor from damaging. A reflective object sensor (OPB0AWZ) [32] is a phototransistor. It is used as an optical end-stop. They are installed on both side of the X and Y axis on the frame as shown in fig. 4.2. These optical sensors are connected to the end-stop section on the Ramps board as shown in fig. 4.3. This installation helps in determining the homing of the motors and determining the zero working coordinates of the experiments.

There are three connectors for each of the end-stop as shown in fig. 4.2. The orange wire which functions as an anode of this sensor is connected to the positive section on the end-stop section on the Ramps board, the green and blue wires which function as the cathode and the emitter of the sensor respectively are clubbed together and are connected to the negative section. Finally, the white wire which is the collector part of this sensor is connected to the signal section.

### **Software:**

The final main component of the control electronics is software. Communication is established with both the controllers, the stepper motor controller and the dispensing driver controller, using a Java Simple Serial Connector (jssc). This is an open source free java library. This communication is done using a serial port communication. The mechanism to achieve this process is discussed in detail in section 4.3. The firmware installed on the motor controller to assist the motor movements are discussed in below section.

#### **4.2.2 Software for Motor Controller**

The first step to execute the printing process is to create a structure and converts that into instructions that motor controller understand. A java application program is designed to accomplish this task. The design of this program is discussed in detail in section 4.3.

The next step is to install the firmware on the motor controller board. This firmware or program is used to facilitate the motor movements. This interprets the instructions and send the actual electronics signal to the motor to move.

Marlin is an open source firmware that is commonly used for the Ramps motor controller. This is used as the firmware for this system. It runs on the Arduino board and is compiled and uploaded on the motor controller. It includes the configuration of the motors. The Marlin program is edited to calibrate the axis and set the configuration of the motor. The setting of configuration of the stepper motors is described in below section. Marlin is edited to execute the motor movements, manage and notify the user about all the real-time activities of the machine such as the status of the end-stops and the details of any printing exception. The details of printing errors, and the ways to handle them is described in detail in section 4.3.

#### **4.2.2.1 Setting Configuration of The Stepper Motor**

Marlin is edited before uploading it to the motor controller. The communication preferences such as baud rate is set to the program. The motor controller board will accept transmissions over the USB connection at this rate. “Configuration.h” file of the Marlin is edited to achieve the same. Baud rate is in the “Configuration.h” file of the Marlin and is defined as (`#define BAUDRATE 250000`). This baud rate later is implemented as the communication preferences in java to set up the communication with this controller board.

The direction of the movement of the motors is already discussed earlier. The configuration of the stepper motors in the Marlin is edited to achieve the same. Homing is adjusted in the Marlin. This makes firmware moves all required axis until it hits the end-stops. The direction of the end-stops while homing is adjusted in the “Configuration.h” file of the Marlin. They are stored in

variables “X\_HOME\_DIR”, “Y\_HOME\_DIR” and “Z\_HOME\_DIR” for X, Y and Z axis. The homing direction is X\_MIN and Y\_MIN for X and Y axis respectively for this work.

Working zero coordinates are also manually adjusted in the same file. After sending the homing command to the motor controller, the motor moves to the required set homing position and comes back to the working zero coordinates position. The distance from the end-stops to the working zero coordinates is measured beforehand using a metric scale and set in the “Configuration.h” file in the Marlin.

#### **4.2.2.2 Axis Calibration**

The calibrated steps per unit in mm is set in Marlin for all the axes. The stepper motor is of 0.9° step angle. This is 400 steps per revolution. The configured micro stepping of the installed Pololu driver with jumpers is 1/16 (obtained from datasheet) [33]. This means that for 1/16 micro stepping resolution, the steps per revolution becomes ( $400 * 16 = 6400$  steps / revolution).

Also, the mechanical components influence the resolution per unit. Thus, the above calculated value of steps / revolution changes. An 8 mm Metric Acme Lead Screw is used for all the three axes for this work. The above calculated value of steps per revolution becomes ( $6400 \text{ (steps/revolution)} / 8 \text{ (mm/revolution)} = 800 \text{ steps/mm}$ ). This value is set as default steps per unit in mm for each of the three axes. The axes are calibrated as:

X, Y-Axis:  $50 / [\text{measured length in mm}] \times [\text{current steps/mm}]$

Z-Axis:  $10 / [\text{measured height in mm}] \times [\text{current steps/mm}]$ .

The X axis is asked to move 50 mm with set 800 steps/mm in Marlin and the distance moved is noted. The process was repeated till the measured movement value was same as that of the required set movement value. These values were set in the “Configuration.h” file of the Marlin

and stored in the variable “DEFAULT\_AXIS\_STEPS\_PER\_UNIT”. The calculated measured value for X, Y and Z axis were 797.40, 799.14, 817.94 steps/mm respectively.

All these calibrated values are set in Marlin program. The program is now uploaded to the motor controller using Arduino IDE. The motor controller is ready to execute the motor movements. The motor and the dispenser controller are communicated and controlled with the host computer to execute the printing tasks. The host computer runs software for this purpose. The below section deals with the design of the software for the same.

### 4.3 System Architecture:

The host computer runs the software program. This program is designed to parse, execute and transmit user command and offer interactive user interface to control the inkjet printing system using the host computer. The system architecture designed are broadly classified into two layers. They are, (i) Interactive User Interface layer and (ii) Control layer.

Figure 4.4 depicts the system architecture. It shows the integration of the main modules of the inkjet printing system with the host computer. This is done to execute the printing process. Section 4.3.1 discusses the design and implementation of the user interface layer on the host computer and section 4.3.2 deals with the control layer. This layer includes the design mechanism to control and interact with the main components of the inkjet printing system (motor controller and dispenser electronics controller) using the instructions or command.

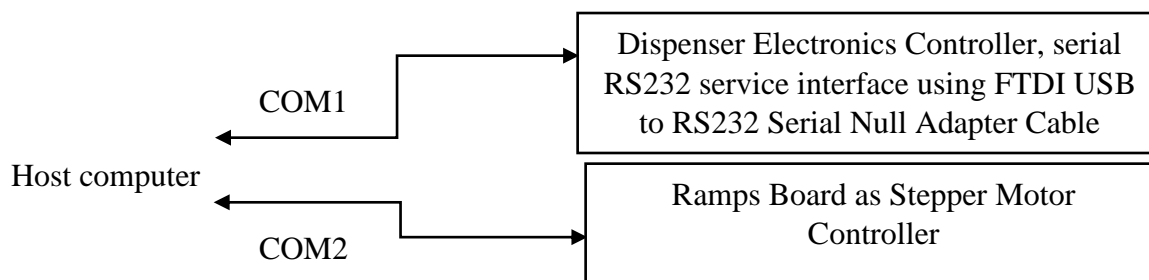


Fig. 4.4: System architecture. Host computer interact and control the controllers.

#### **4.3.1. User interface layer**

This allows user to directly interact with the inkjet printing system to execute the printing tasks. The main modules of the inkjet printing system are, (i) The motor controller and (ii) The dispenser electronics controller. The designed user interface layer is shown in fig. 4.5. This interface serves the following purpose:

- Users can query the real time running state of the motor and can control the motor and nozzle controllers to execute the printing tasks.
- Users can gather the current state of sensors such as optical and mechanical end stops installed on the frame. The assembly of frame is described in chapter 3.
- Users also get notified about details of any printing exception and emergency in real time. They know the source of the error and can rectify that and start the printing process again.
- Users can select any of the three design patterns for printing process. The patterns are, (i) User can create a line in X, Y plane, (ii) User can design a rectangle in X, Y plane, (iii) User can design a circle, and (iv) User can create a dot at X, Y plane. from the user interface
- Users also have the option to provide desired drop spacing value in mm.

The program converts these designs in a machine friendly instructions or command to execute the printing tasks. The implementation of the program design is discussed below.

##### **4.3.1.1 Backend implementation of Program design**

A java application is designed to establish the communication and control the inkjet printing systems. A java simple serial connector or jssc jar is used for communicating via serial port communication. It is an open source free java jar or library. This library is included in the application. The main features that it provides are discussed as follows:

- The library has functions to assist the user to get the list of available COM ports and set the communication preferences such as baud rate using “setParams ()” method to the ports. This is done to establish the communication.
- It also provides features to send and receive the data through the ports. The data is sent or received in the form of java “String”. This process is done using “writeString ()” and “readString ()” methods to and from the serial port.
- It provides a listener interface and a “serialEvent ()” function. A class is created to implement that interface and override that function. The listener is added to the connected port. A proper implementation of this listener is done to notify the java application if any data is sent or received to the connected port. Code can be referred in Appendix D.
- This library is very well supported in windows 32 and 64 systems, Linux, Solaris and Mac. It is designed to operate in a multi-threaded system.

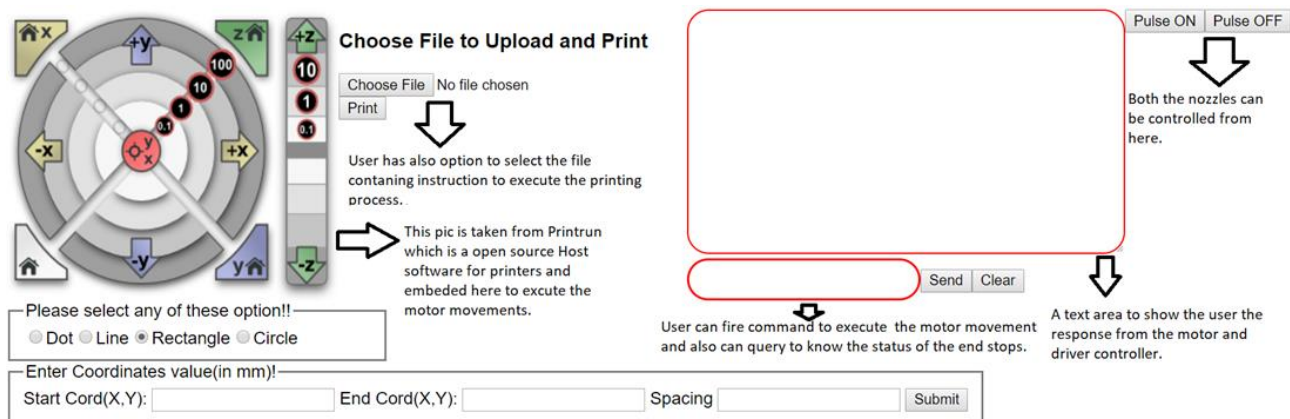


Fig. 4.5: User interface designed in Java.

The application is run and deployed using NetBeans IDE. The apache tomcat is used as server to run the designed application. There are two ports connected with the host computer. One of the ports is for motor controller and the other is for dispenser electronics. The connections to these ports are established while deploying the application. Each of the two ports are initialized and communication preferences (baud rate, data bits, stop bits, parity) are set as follows:

- Motor positioning control electronics via a USB to serial communication with baud rate as 250,000 bits/s.
- Dispenser electronics via a serial RS232 interface with the following setting (i) Baud rate: 19200 bits/s, (ii) Data format: 8-bit, (iii) Stop bit: 1, (iv) Parity: none.

Java server pages (JSP) is used to design the user interface page. A JavaScript and jQuery are also used to populate the result and to design the page. This is useful in following ways:

- It helps the developer to create both the static and dynamic content.

The dynamic content here, is to notify user about the real time activities of the inkjet printing system such as any printing exceptions, the status of the end stops or the current state of motor movement execution. The dynamic content is displayed in the text area designed in user interface page. This is shown in fig. 4.5.

User select the design pattern and enters the start, end (X and Y) coordinates value and the drop spacing value in the text box provided. The values entered are taken in mm. User click on the submit button to start the printing process. Each of the patterns are discussed in detail below:

### **1. A line**

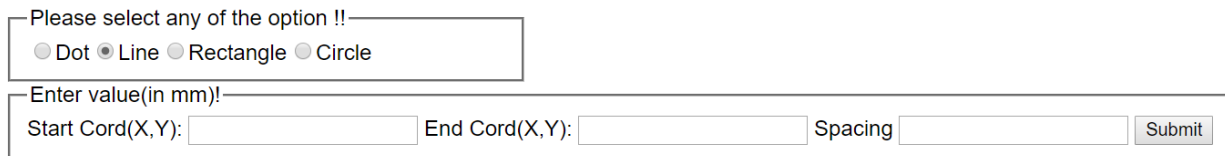
Figure 4.6 shows selecting a line on the user interface. User enter the “Start Coord (X, Y)” and “End Coord (X, Y)” of the line. The values entered are stored in the four variables “x\_start”,



“x\_end”, “y\_start” and “y\_end” and the tangent of the line with x and y start and end coordinate values is calculated as,

$$\text{float angle} = (\text{float}) \text{Math.atan2}(y\_end - y\_start, x\_end - x\_start);$$

The entire line is now divided into smaller segments of X and Y coordinates based on the drop spacing value entered by the user and the angle calculated above. They are created as,

$$x\_start = x\_start + \text{drop spacing} \times (\text{float}) \text{Math.cos}(\text{angle});$$
$$y\_start = y\_start + \text{drop spacing} \times (\text{float}) \text{Math.sin}(\text{angle});$$


The figure shows a user interface with two main sections. The top section is a box containing the text "Please select any of the option !!" followed by four radio button options: "Dot", "Line", "Rectangle", and "Circle". The "Line" option is selected. The bottom section is a larger box containing the text "Enter value(in mm)!" followed by three input fields: "Start Cord(X,Y):", "End Cord(X,Y):", and "Spacing". Each input field has a small text box next to it. To the right of the "Spacing" input field is a "Submit" button.

Fig. 4.6: Option to select a Line on user interface page.

These values are formatted to two decimal values. The calculations are done in a loop and the loop is continued till smaller segments reaches the “End Coord (X, Y)” values. User enters the end coordinate values on the user interface. This forms many small line segments. The entire line with start and end coordinates, is now divided into smaller line segments. For example, if user wants to draw a line starting with 0,0 and ends at 10,0 (mm) with drop spacing value as 100  $\mu\text{m}$ . The entire 10 mm line is divided into 100 smaller segments. So, a total of 101 smaller segments are created including the starting 0,0 coordinate value. Each of these smaller segments are created with x and y coordinates.

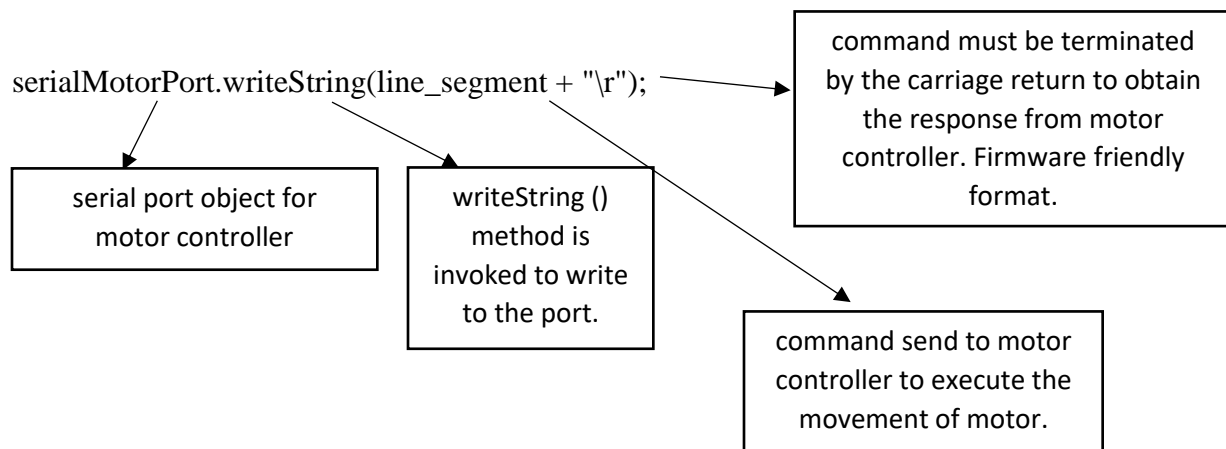
The coordinate values of these line segment are arranged in a friendly format and are stored and saved in a txt file. Each of the coordinate values of the line segments are the instructions or

command. The instructions must be terminated by a newline. This is done to receive the correct response from the controller. Instruction are stored line by line in a file to form a friendly format instruction. The format is shown below:

```
"G1" + " " + "X" + x_start + "Y" + y_start + "\n";
```

Here, “G1” is the function. This function is implemented in Marlin firmware to execute the linear movement. “X” and “Y”, as shown above, are the planes of movement and “x\_start” and “y\_start” are the parameters of the movement in X and Y planes respectively. The code is added in appendix A.

The file is saved at any location in host computer and each line of the file is read and sent via a serial port to the motor controller to execute the movement. This process is achieved as shown below:



Here, the “writeString ()” function of the “SerialPort” class (feature provided by the jssc jar or library), is to facilitate the sending of the instruction in the “String” format. Each command is terminated by a carriage return. Finally, the correct format which is stored in the above “line\_segment” variable is shown below:

```
G1 X10.0 Y0.0
```

Here, G1 is the function invoked for linear movement. X and Y are the plane of the movement and the 10.0 and 0 are the parameters for the movement in mm. This instruction will create a 10 mm line in X, Y plane starting with 0,0 coordinate value. Marlin firmware installed on the motor controller receives this instruction and operates in following ways:

- It receives the above movement command and add them in a queue and execute them in the order they added.
- It parses the received data to retrieve the command parameters. During parsing, it validates the incoming command in sequential order and reports an error message for incorrect command. The handling of errors for this work is discussed in section 4.3.2.1.
- Lastly, the queue is being processed, converting linear movements into precisely-timed electronic pulses to the stepper motors.
- An “ok” message is send as response for each correct movement [34].

The next structure is a rectangle. The backend implementation of this design pattern is discussed below.

## **2. A filled rectangle**

The user selects the rectangle option and enters the start, end (X, Y) coordinates value and drop spacing in mm in the text boxes provided and click on the submit button as shown in fig. 4.7. Here, the start coordinates refers to the starting coordinate points where user wants to start to print a rectangle, whereas the end coordinates refers to the ending coordinates points. For example, for a (1×1) rectangle starting at (0,0), the start coordinate refers to (0,0) and end coordinate refers to (1,1) printing a desired length and width of 1 mm rectangular structure.

Please select any of these option!!

☐ Dot ☐ Line ☒ Rectangle ☐ Circle

Enter Coordinates value(in mm)!

Start Cord(X,Y):  End Cord(X,Y):  Spacing

Fig. 4.7: Option to select rectangle as a design pattern on user interface page.

Here also, the coordinates of a rectangle entered by the user are stored in the four variables like that of a line. The process follows the same procedure of the line structure. The line is divided into smaller segments based on the drop spacing value in mm. Code is added in appendix B.

Here, parallel horizontal lines are created in a way that the end coordinates of one entire line becomes the start point for the second parallel horizontal line. The second line is started just above where the first line ends. The distance between each of the parallel horizontal line is the drop spacing value entered by the user. The line ends until it reaches the end coordinate values entered above. In this way, a rectangle is created and filled. The instructions are created in the same format as discussed above and sent to the motor controller using serial communication.

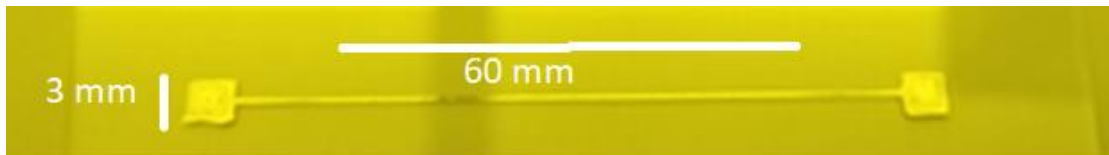


Fig. 4.8: A 3×3 mm rectangle pads with a 60 mm line created and sintered on a glass substrate.

The next structure is that of a circle which is discussed below. Here, the filling of the structure is done in a different way.

### 3. A filled circle.

User here provide the X and Y centre coordinate of the circle. These values are stored in two float variables. The radius of the circle is stored in another float variable. A Radio button is provided for clock or anticlockwise start of the circle. User interface page is shown in fig. 4.9. A

“ten\_degree\_point\_spacing” circle is created starting with selected clock or anticlockwise option. If clockwise option is selected, the circle will start creating in a clockwise direction and vice-versa.

X and Y coordinates of the circle are created as

$$\text{circle\_X\_coord} = (\text{circle\_X\_centre} + \text{radius} \times (\text{float}) \text{Math.cos}(\text{angle}));$$

$$\text{circle\_Y\_coord} = (\text{circle\_Y\_centre} + \text{radius} \times (\text{float}) \text{Math.sin}(\text{angle}));$$

Here, angle is at every ten-degree spacing. The above calculated values are appended in the same way with the string “G1” as done for a line and a rectangle and transmitted via serial port to the motor controller.

Filling of the circle is done with the vertical lines. Each of the vertical lines are separated with the drop spacing value. The line is drawn in between two X extremes points. These extremes are the X coordinates of the two ends of the circle diameter at an angle zero. The drop spacing value in mm is represented as “Spacing” in below figure. The Y extremes of the vertical lines are at equidistance from the centre coordinates.

Please select any of the pattern to print !!

☐ Dot
☐ Line
☐ Rectangle
☒ Circle

Enter Coordinates value(in mm)!

Center(X,Y): 
Radius: 
Spacing: 
☒ Clockwise
☐ Anti-Clockwise

Fig. 4.9: Option to select circle as a design pattern on user interface page

Here the movement instructions are created in the same format as discussed above while creating a line and a rectangle. Code is shown in appendix C.

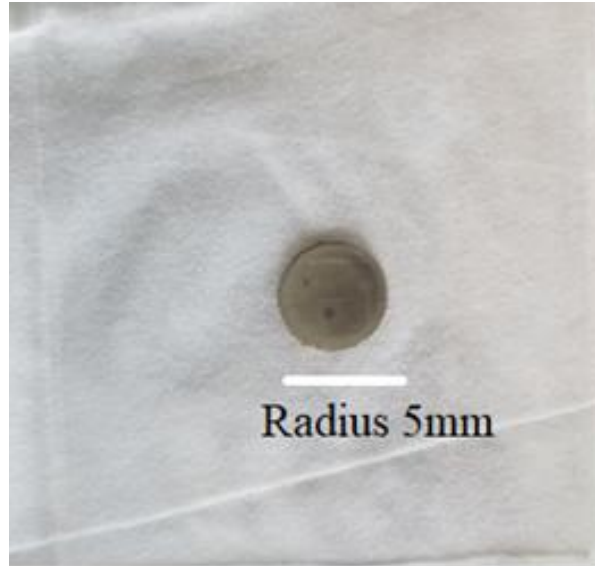


Fig. 4.10: A filled circle of radius 5mm created and sintered on a glass substrate.

#### 4.3.1.2 File Upload Feature

Instead of selecting the design structure, user also has the option to write the structure with the coordinates value in a file and upload that file to the user interface page. There is also a file upload feature on the user interface. The structures are written in following format in the file and uploaded to the user interface.

- **Format written in a file to create a vertical or a horizontal line**

L: Start Coordinates (X, Y) in mm: End Coordinates in mm: Drop spacing value in mm

- **Format written in a file to create a filled rectangle**

R: Start Coordinates (X, Y) in mm: End Coordinates in mm: Drop spacing value in mm

- **Format to create a filled circle**

C: Centre Coordinates (X, Y) in mm: Radius in mm: Drop spacing value in mm:

Clockwise/Anticlockwise

A print button is provided on the user interface next to the upload file feature. It is clicked to print the structure on the substrate. All these structures are drawn as discussed above.

#### **4.3.2. Control layer**

This layer discusses the mechanism of the interaction with the main modules of the inkjet printing system. The interaction with the modules is described as follows:

##### **1. Interaction between host computer and motor positioning control electronics.**

The interaction is described in following steps:

- The application is run and deployed in NetBeans IDE. Two serial ports are connected while deploying the application. One port relates to the motor controller and the other port is connected with the dispenser controller. This connection establishes a communication between the host computer and the controllers.
- User selects any of the four patterns option and enter the start, end (X, Y) coordinate values and drop spacing value in mm in the text box provided on the user interface page.
- The design of these patterns has already been discussed above. The patterns are created and arranged in a friendly instruction and saved in a txt file.
- The first line of that file is read and send to the motor controller via a serial port. Each line of the file is a movement command in correct format.
- Motor controller sends an “ok” message in response to the correct command. The motor controller is ready for the next command to execute.
- The application waits until “ok” message is received from the motor controller.
- The application proceeds thereafter to interact with the dispenser controller. The communication with dispenser controller is discussed later in this chapter.
- The dispenser controller responds properly if nozzle dispenses the ink drop.

- The application waits again until proper response from the dispenser controller is received.
- Thereafter, the next line of the file is read. This means the next movement command is sent.

In this way, one drop is dispensed for every movement. Every movement is created based on the drop spacing value. Hence, for every drop spacing value, nozzle dispense one ink drop. The above process is repeated until the last line of the file is read and a pattern is created. The entire process is described in below flowchart.

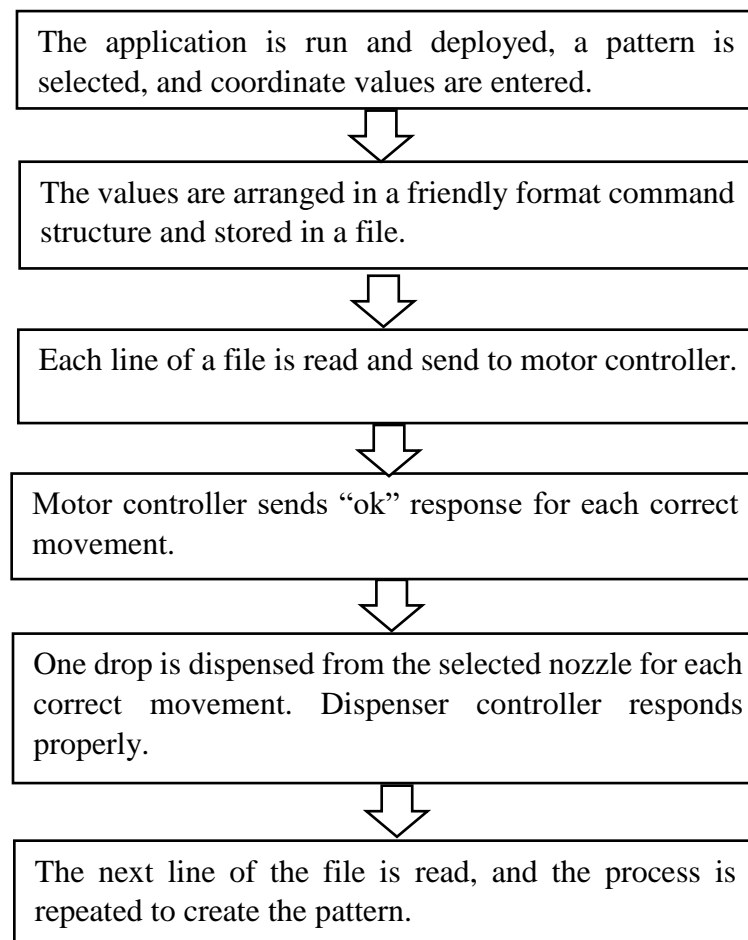


Fig. 4.11: A flowchart mainly to understand the interaction with the motor controller.

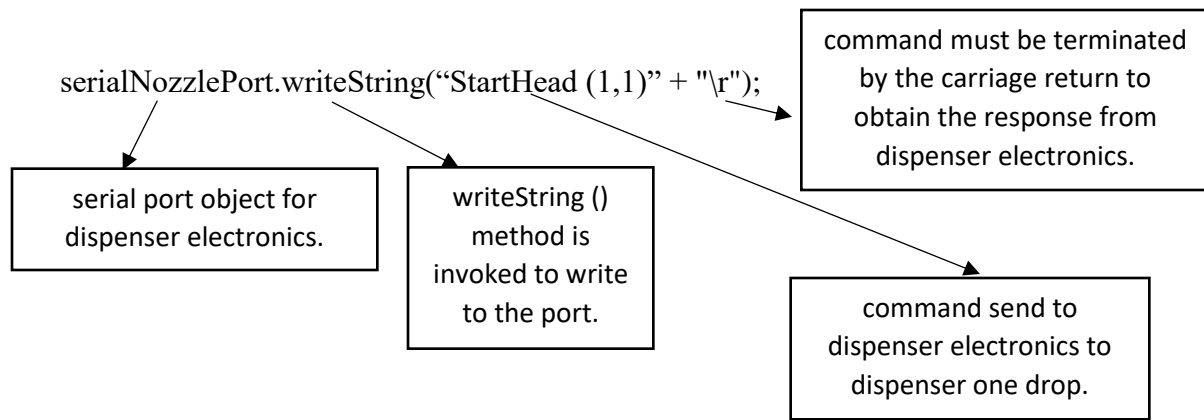


The complete interaction of the motor and dispenser controllers with the host computer is discussed below.

## **2. Interaction between user and micro dispenser driver electronics.**

This interaction starts only when “ok” success message is received from the motor controller. The interaction can be described in following steps.

- a) The dispensing parameters such as voltage and pulse width are pre-set before starting the application. These parameters are required to dispense one ink drop from the nozzle. Only one nozzle is used for this research work, as discussed already in chapter 3.
- b) The application is run and deployed to set up communication between the host computer and the controllers.
- c) User selects the design on the user interface and enter the start, end coordinates and drop spacing value in mm and click in submit button.
- d) The design is created and stored in a file. This is already described above.
- e) Each line of the file is read and send to motor controller. Motor controller sends the “ok” message for each correct movement execution.
- f) Nozzle 1 is started to dispense one drop based on the pre-set parameters.
- g) For each correct movement, one drop is dispensed from the nozzle. Nozzle Head is started using “StartHead (1,1)” command to dispense one ink drop. This command is sent to the dispenser electronics via the connected serial port. The format of sending the command is shown below. The command is terminated by the carriage return to make it a correct command structure.



- h) Dispenser controller sends “>” message as a prompt response if the nozzle dispenses correctly.
- i) The application waits until it receives this message. This message is displayed on the user interface to confirm that dispenser has responded properly and is ready to receive another command.
- j) Thereafter, the nozzle is closed. The close command to stop the nozzle is “StartHead (1,0)”. This command is also set in the same format, as discussed above and is sent to the driver controller via a serial port.
- k) Again, the application waits again till it receives “>” message to make sure that the nozzle is closed properly.
- l) Dispenser controller sends “?” message if it doesn’t dispense properly. This message is also displayed to user interface. The two connected serial ports are closed, and the complete process is stopped. User must start the application again to reconnect the system.
- m) If the nozzle is closed properly after dispensing one drop, the next line of the file is read and is sent to the motor controller to execute the next motor movement.
- n) This process is repeated till the last line of the file is read and a pattern is created.

The below flowchart shows the interaction with the controllers.

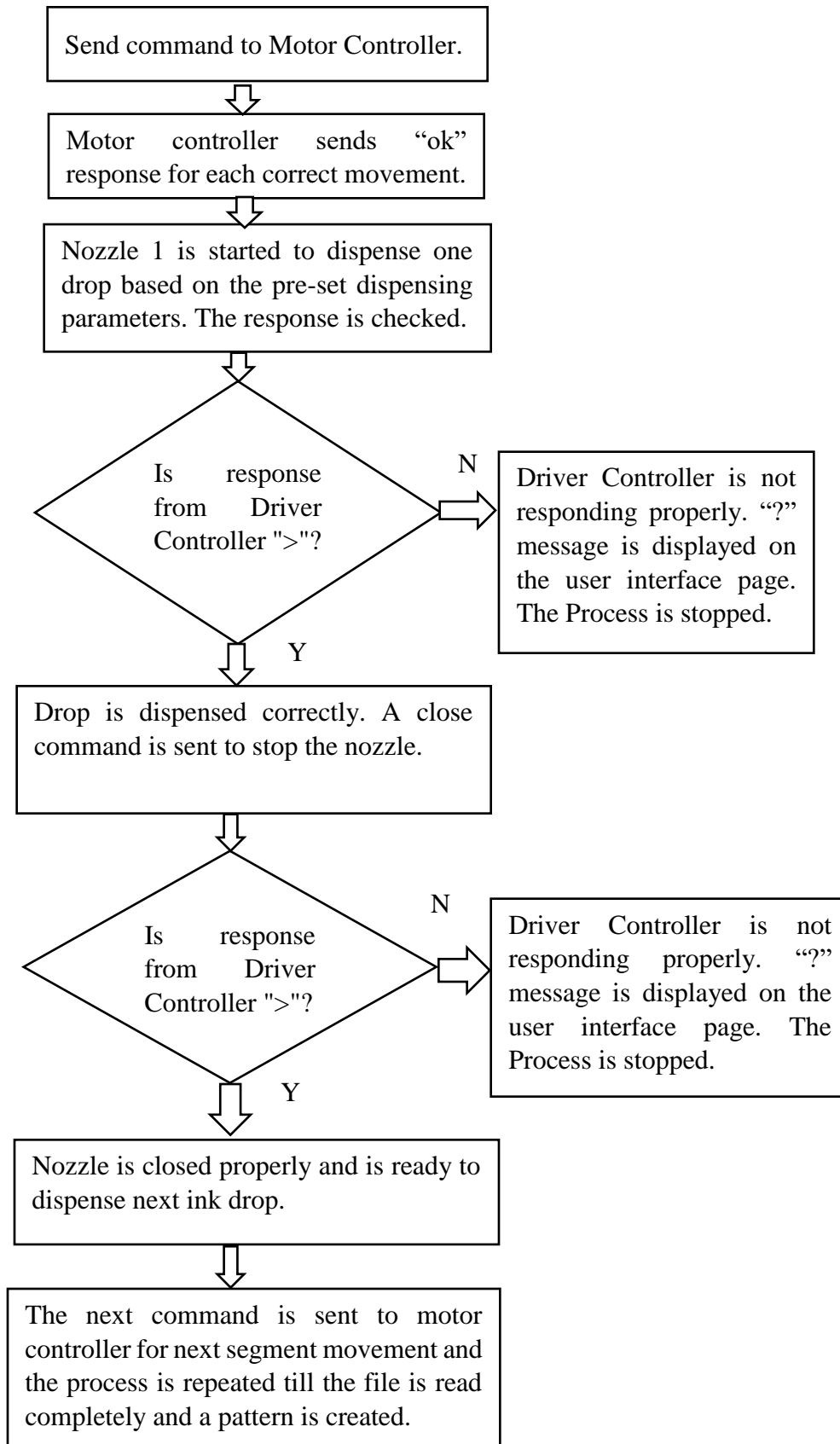


Fig. 4.12: A flowchart to understand the interaction with the motor and dispenser controller

The errors and exceptions may occur while communicating with them. These exceptions are handled properly. They are discussed in following section.

#### **4.3.2.1 Handling Error/Exception from the Controllers**

The motor and dispenser controllers send the error message to the host computer if any error or exception is encountered. The error or exception occurs if any unknown command or an incorrect command format is sent to the motor controller or an emergency stop situation arises. Emergency stop situation is discussed later in this chapter. The motor controller detects that error while parsing the instructions and send the error response message to the java application. The application displays the message in the text area on the user interface page. This helps to notify the user about the error.

The different errors that arises with the motor controller are discussed below. The motor controller sends “busy: processing” message if the motor execution is not completed and the next command is sent. This message is displayed on the user interface. It notifies the user that the motor execution is not completed. The controller sends “ok” message once the movement is completed.

- **Unknown Command error message:**

This error message is received from the motor controller when an incorrect command is sent to motor controller. If user sends an incorrect command to query the details of printing execution such as the state of the sensor attached or the coordinates of the motor, then this error message is displayed.

A text box is designed to send the command from the user interface page and a text area is provided to view the response from the motor controller. This is shown in fig. 4.14. The user can send the correct command again. The controller will respond “ok” for correct command execution.

This error message is not encountered while printing as the command are stored and saved in correct format in the file.

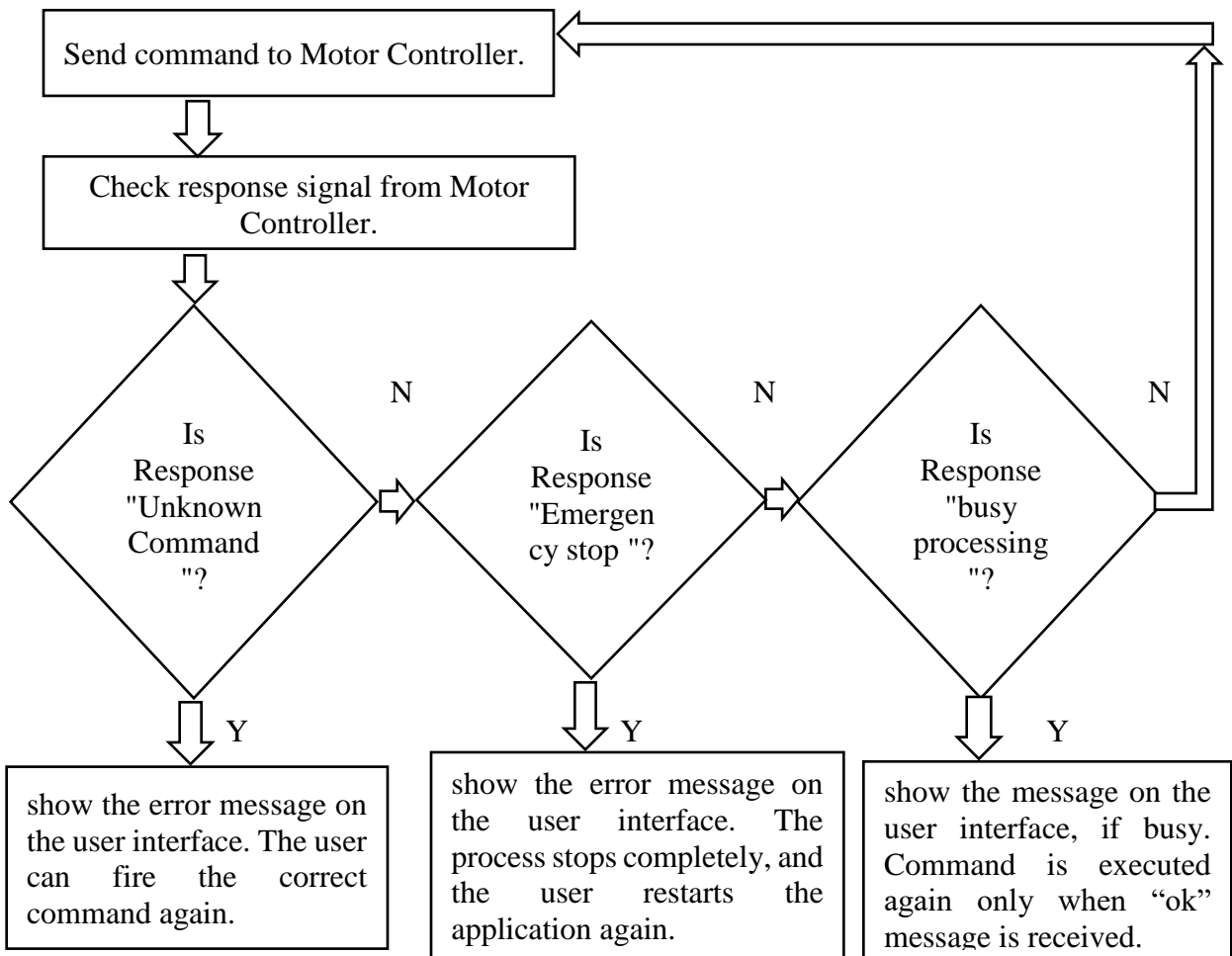


Fig. 4.13: A flowchart to understand the interaction with the motor controller while handling errors.

```
jssc.SerialPortException: Port name -  
COM4; Method name - openPort();  
Exception type - Port not found.
```



A response shown if the port is not connected.

User can fire command to execute the motor movement and can also query the status of the endstops.

(a)

```
start  
echo:Marlin 1.1.1  
  
echo: Last Updated: 2017-05-12 12:00 |  
Author: (none, default config)  
Compiled: Nov 15 2018
```

Response from motor controller is connected properly.

(b)

```
Head has started  
Nozzle Response is>  
Nozzle responded Properly, Nozzle OFF  
Move segment by segment G1 X5.0Y0.0  
motorResponse is ok  
  
Head has Started  
Nozzle Response is>  
Nozzle responded Properly, Nozzle OFF  
file is finished
```

Response shown when printing ends.

(c)

Fig. 4.14: A text area on user interface showing the response to notify the user, (a) A response obtained if the ports are not closed, (b) A response obtained showing the motor controller responded properly, (c) Response when printing ends.

- **Emergency stop error message:**

This situation arises if the optical end stop attached to the X and Y axes stops working. If this stop working, the stepper motor hits the mechanical switch. The mechanical switch is attached

to both ends of X and Y axis along with the optical end stops. The assembly of mechanical switch with the optical end-stop on the X axis is shown in fig. 4.2.

A 5V, two-channel relay module interface board is used to control this situation. A 12 V power is required to operate the motor controller board. This power is controlled using this relay. The relay is equipped for high-current relay, AC250V 10A and DC30V 10A. This situation is depicted in below fig. 4.15.

In the normal condition, relay is on and the output of the relay keeps the motor controller on. The required 12 V is supplied to the motor controller. The “12V IN” in the fig. 4.15 shows the same. All the four mechanical switches, two each for X and Y axis, are connected in series. This goes to the input of the 2-channel relay. Now, if any of the mechanical switch is hit, the input drives the relay to off. This cuts off the power supply to the motor controller and the printing execution stops.

The other channel of the relay drives the pin on the motor controller board. This pin is located besides the reset button. This pin is connected to PIN\_12 on the Arduino in Marlin configuration. Arduino Mega pin assignments with the Ramps board is mentioned in “pin\_Ramps.h” file. Marlin installs on the motor controller, is edited to check the status of this pin. The status of the pin is checked in the “Marlin\_main.cpp” file of the Marlin firmware. If the relay is off, the status of this pin becomes “PS\_ON\_ASLEEP” and an “Emergency stop” message is sent to the host computer via the serial communication to notify the user about the situation.

The status of this pin is always checked once the power is on and printer is ready, and communication is established. The message is sent using “SERIAL\_PROTOCOLLNPGM (“EMERGENCY STOP”)” via a connected serial port. Here, “EMERGENCY STOP” is the message. Once the host computer receives this message, the java application communicates with

the motor controller and calls the function of the Marlin firmware to disable all the motors. The application also communicates with the dispenser controller to stop the nozzle to dispense ink drop. Both the serial ports are closed. The entire process comes to halt. User restart the application again.

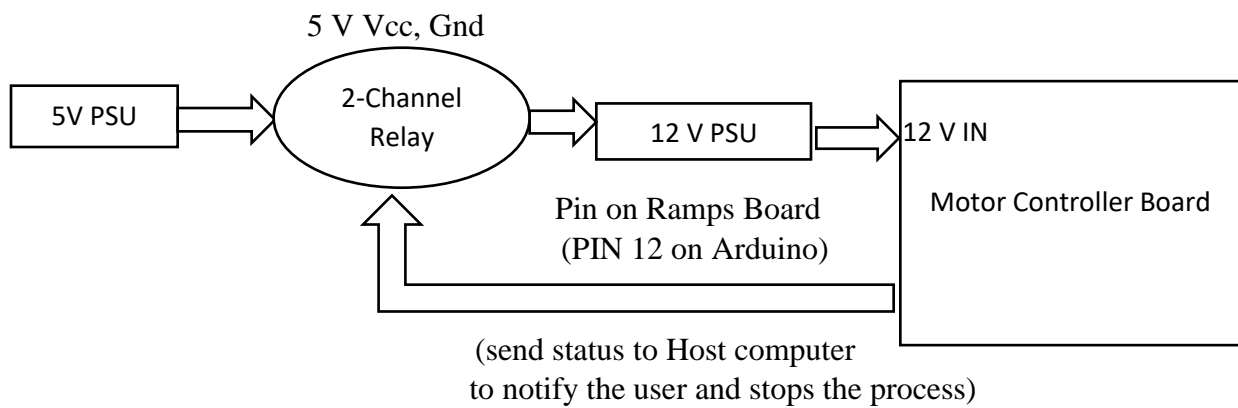


Fig. 4.15: Emergency stop design implementation



## **Chapter 5: Fabrication of Conductive Lines using Silver Nanoparticles and Sintering**

### **5.1 Introduction**

The inkjet printing process involves the ejection of a fixed quantity of a liquid phase material, usually called ink. This material forms the main components in the inkjet printing process. This chapter deals with the fabrication of the conductive lines using the metallic nanoparticles suspension.

We have already discussed that printing of the flexible electronics using drop-on-demand inkjet printing system offers many advantages. The size of the printed features reduces which decreases the production costs of electronic devices. The ink drops are dispensed and deposited on the substrate on demand. This reduces the usage of material or ink [35].

The metallic nanoparticles have gained more importance these days as it provides better conductivity than any others conductive polymers. These particles can easily be operated at room temperature. Gold and silver nanoparticles are widely used as the conductive ink due to the high conductivity and high thermal stability. The synthesis of silver nanoparticles is much simpler [36].

These particles have been used in many applications. Some of the applications are interconnections for a circuitry on a printed circuit board, disposable displays and radio frequency identification (RFID) tags and organic thin-film transistors [37].

Silver nanoparticles are used as the conductive ink for this research work. Section 5.2 deals the conductive ink and its features. After printing the silver particles, thermal processing of the printed structure is required. This is required to form the connections between neighbouring molecules and form the highly conductive features. This process is known as sintering [38]. This chapter deals with the detailed discussion of this mechanism. Oven sintering is the most common

sintering process. In this thesis both oven sintering and heating by an incandescent lamp are used. Section 5.3.1 discusses the result obtained using oven sintering process, and section 5.3.2 deals with the results obtained using an incandescent lamp sintering process.

## **5.2 Printing of Silver Nano-particle Conductive Ink**

The main characteristics of interest for the conductive ink are properties such as viscosity and surface tension. The viscosity determines the form and the volume of the dispensed ink drops and surface tension influences the velocity of the ink drops tail and breaking of the ink drops. This also determines the formation of satellite ink drops.

The size of the particles, dispersion and stability of the silver nanoparticles plays a vital role to prevent nozzle from clogging. The smaller particles size is always helpful. In addition, the physical properties of the silver nanoparticles effect the performance of the printed structure [39]. Conductive materials used for inkjet printing system are either solution-based or particle based.

UTDAgIJ Conductive Silver Nano ink, supplied by UT Dots, Inc. was used as a silver conductive ink for this work. Key important properties of the ink are shown in table 5.1. This ink can also be stored for almost more than three months at room temperature without change in its performance. The storage must be kept air tight to prevent the evaporation of the solvents [40]. The experiments were carried out using this conductive ink. The results are discussed later.

Table 5.1: Key important properties of silver nanoparticles ink from the datasheet.

Particle size	~ 8 - 20 nm
Silver concentration	~ 40wt%, ~ 60% w/v
Solvent	Hydrocarbon
Surfactant	Oleylamine
Viscosity	~ 11 cp
Shelf life	3 months at room temperature

The gap between the substrate and the tip of the nozzle plays a vital role in influencing the droplet impact and spreading. This gap is required to stabilise the ink drop. The larger gap between them results in more deviation of the ink drop path from the straight line. This may hamper the drop positioning accuracy and results in longer exposure time of drop to the atmospheric conditions such as temperature and humidity [36]. The gap between the substrate and the tip of the nozzle was maintained at 1mm for experiments purposes. Only a single layer printing is done.

### 5.2.1 Droplet Substrate Interaction

The size of the ink drops varies with the ink-substrate interaction. The size of the ink drops on the substrate defines the line width achieved. The ink drops fall on the substrate under the gravity force and air resistance. The impact of ink drops on the substrate forms its pattern. The solvent present in the ink evaporates partially during the flight from the nozzle to the substrate. The rate mainly depends on the ambient temperature and the vapour pressure of the solvent.

The ink drops either spread or ball up due to high surface tension after falling on the substrate. The surface tension tries to keep the surface area low. The relationship between the

substrate surface energy and the ink surface tension determines the wettability of the ink. A good wetting behavior results in excellent spreading of the ink drops on the substrate [41] [42].

Commercially available microscope glass slides of  $2 \times 3$  inch and 1 millimeter thick were used as the substrates. Surface preparation of the substrate was done before the printing process. This was done to improve the interfacial adhesion between the substrate and the printed structure and remove the dust particles, if any. The glass substrates were thoroughly cleaned by washing subsequently with DI water, acetone and isopropanol. Each side of the glass substrate was rinsed three times. It was dried under the flow of nitrogen and was kept for few hours before using them for printing structures. The same process was maintained for all the experiments.

### Dispensing Parameters

The first set of experiment was carried out for  $100\ \mu\text{m}$  drop spacing. Here, the nozzle and substrate are at room temperature. Table 5.2 summarises the dispensing parameters to dispense UTDAgIJ Conductive Silver Nano inks.

Six parallel lines of length from 10 mm to 60 mm with  $1 \times 1$  mm rectangular pads at both ends were inkjet printed on a single glass substrate. The line width was measured after printing using the Alpha Step profiler. The printed structure is shown below in fig. 5.1. These lines were 3 mm apart from each other. Good quality lines were obtained at  $100\ \mu\text{m}$  drop spacing.

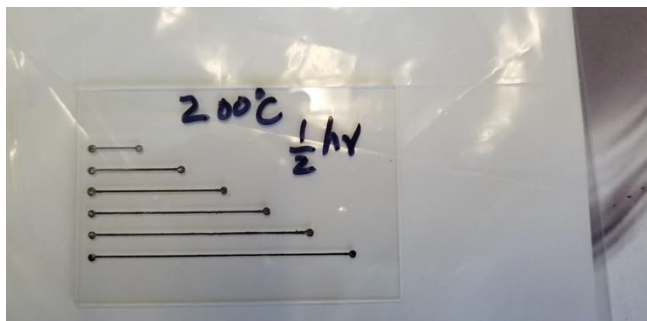


Fig. 5.1: A set of six parallel lines separated by 3 mm printed on a glass substrate and sintered at  $200\ ^\circ\text{C}$  for 30 min.

Ink drops were dispensed at 150 V and 30  $\mu$ s pulse width. A total of 101 ink drops were dispensed for printing a 10 mm line with a 100  $\mu$ m drop spacing. The lines were created with 1 x 1 mm rectangular pads at both ends. These pads were created with the same drop spacing of 100  $\mu$ m as that of a line. A total of 111 ink drops were dispensed for creating a 1 x 1 mm rectangular pad, so the two rectangular pads required 222 ink drops. Hence, a total of 323 ink drops were dispensed to print a 10 mm line with 1 x 1 mm rectangular pads at both ends with a 100  $\mu$ m drop spacing.

Table 5.2: Dispensing Parameters to dispenser silver nanoparticle ink drop.

Parameters	Value	Method of setting parameters
Voltage	150 V	Using Control key of the touch screen of Driver Electronics
Pulse width	30 $\mu$ s	Using Control key of the touch screen of Driver Electronics
Nozzle Temperature	23 $^{\circ}$ C	Using Control key of the touch screen of Driver Electronics
Drop Spacing	100 $\mu$ m	User interface page
Frequency	100 Hz	Using Control key of the touch screen of Driver Electronics
Drops	1	Using Control key of the touch screen of Driver Electronics
Distance of Nozzle and the substrate	1 mm	Z axis kept constant for experiment activities

Figure 5.2 shows a dispensed UTDAgIJ conductive silver nano ink drop generated using a Microdrop MD-K-140 dispenser system at 150 V and 30  $\mu$ s pulse width. This picture was captured on a host computer running xiViewer and connected with USB camera MD-O-539-USB (xiQ USB3.0) from XIMEA. The camera was supplied by the Microdrop technologies along with the dispenser system.

The external dimension of the nozzle structure, as shown in figure, was viewed using the connected camera. The external diameter of the nozzle was measured. It was 1.25 mm. This value was made as a reference and was used to calculate the diameter of the dispensed ink drop using ImageJ software. The diameter of the ink drop measured was 95  $\mu$ m.

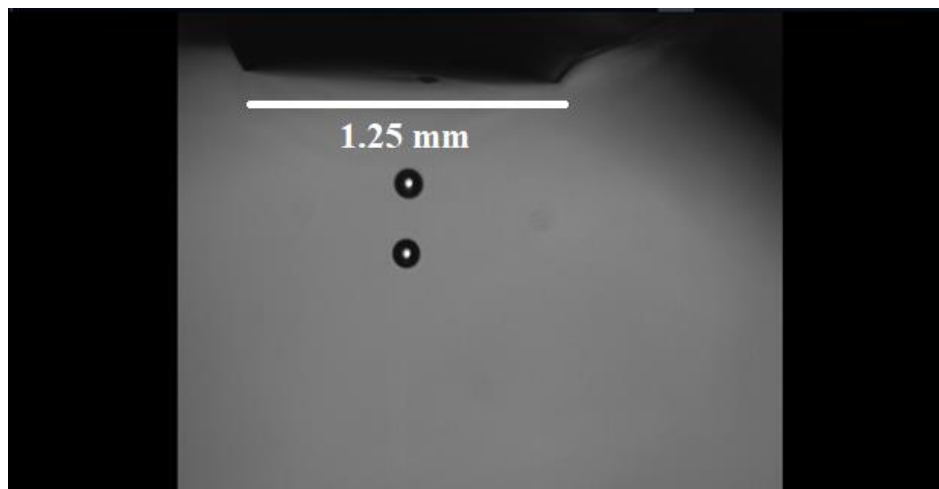


Fig. 5.2: A micro dispensing system from Microdrop Technologies generating a 95  $\mu$ m drop diameter of UTDAgIJ conductive silver nano ink at 150 V, pulse width 30  $\mu$ s and 100 Hz.

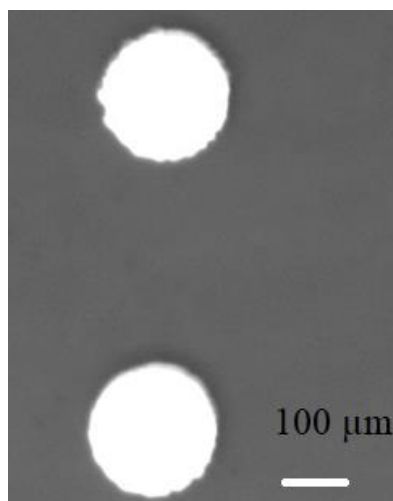


Fig. 5.3: A sintered silver conductive silver nano ink drop. The drop was dispensed at 150 V, pulse width 30  $\mu$ s, viewed using optical microscope.

### 5.3 Sintering Process

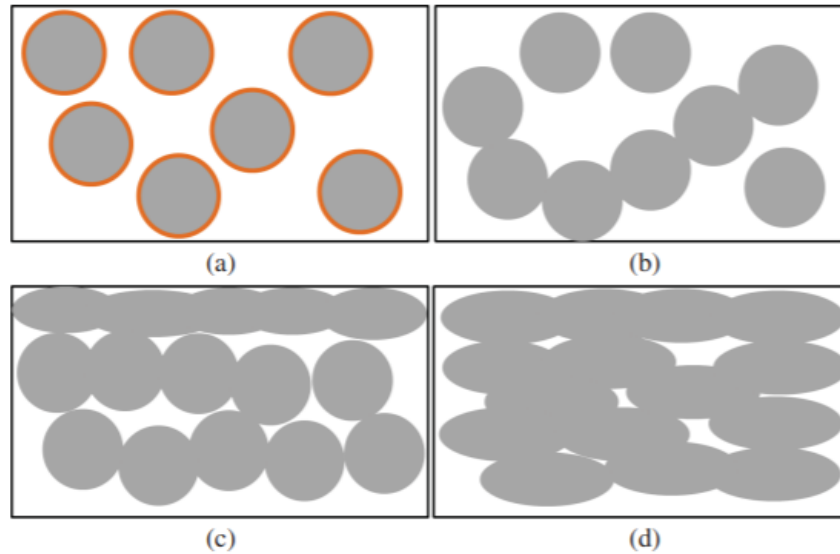
The processing of the printed structure is required to remove the non-metallic component of the deposited ink and give continuity to the structure. This is done to form the conductive patterns. Two properties are very important for the sintering process. The first one is the lowest temperature at which printed features become conductive. This depends on the organic additives in the ink. The second property is obtaining the lowest possible resistance of the printed structure at the lowest possible temperature. UT dots silver nanoparticles ink [40] used for the experiment work suffice both these needs. UT dots silver nanoparticles ink are with an average size around 10 nm and dispersed in a liquid vehicle. Since, it is surface stabilized, this ink is highly soluble in nonpolar organic solvent and can be easily cleaned with acetone, IPA and ethanol. Since, silver is an excellent thermal conductor, the silver spreads heat uniformly by thermal conductance [8].

In such type of nanoparticle-based ink, dispersants prevent the nanoparticles from agglomeration. The nano-silver particles, along with the dispersants submerged in a solvent, are

designed to help initiate sintering at lower temperatures. Sintering, in context of inkjet printed electronics refers to the evaporation of the solvent and the dispersant and the combination of the particle growth and grain-boundary migration. Larger particles are formed due to migration and particle coalescence. Grain boundaries are the places where the orientation of the atomic arrangement changes. The grain-boundary migration is mainly the movement of the boundary separating two grains. The size of the particles may have one or several grains within it. This movement is due to the diffusion of single atom from one grain across the boundary to the other grain.

At the start, single nanoparticles are surrounded by a dispersant. The dispersant keeps the particles apart from each other and no electrical conductivity is formed. Upon heating, the first stage of the sintering process is the evaporation of the solvent and the dispersant from the deposited ink. At this stage, nanoparticles remain discrete. With continued heating, the particles start to coalesce. At the final stage, the deposited ink layer is continuous. This final stage indicates that the particles are in contact to make the continuity of the structure. Higher temperatures are sometimes required to evaporate the dispersants and the solvent to realise a more continuous deposited ink layer and a lower resistivity [42]. However, contaminants sometimes cause discontinuity of the sintered metal. The metal nanoparticles have lower sintering temperatures as compared to the bulk metal melting temperature. This is mainly due to the increased surface-to-volume ratio of the nano-sized metal particles [41] [42].





Reproduced from [I. A. N. Ink, E. Halonen, T. Viiru, K. Östman, A. L. Cabezas, and M. Mäntysalo, "Oven Sintering Process Optimization for Inkjet-Printed Ag Nanoparticle Ink," IEEE Trans. COMPONENTS, Packag. Manuf. Technol., vol. 3, no. February 2013, pp. 350–356, 2012].

Fig. 5.4. Four stages of silver nanoparticle structure during the oven sintering. (a) Single nanoparticles surrounded by a dispersant and submerged in a solvent. (b) Dispersant and solvent evaporates, and conductive paths are forming. (c) and (d) Diffusion and partial melting of nanoparticles progress from top to bottom of the ink layer [43].

Various approaches have been carried out to sinter the metal nanoparticles to form the conductive features. Some of them include thermal sintering by conventional oven heating, exposure to UV radiation, laser sintering, plasma and flash lamp sintering [38]. Thermal sintering in the oven is the most commonly used sintering process.

### 5.3.1 Experiments using Thermal Sintering in the Oven

The sintering process was done at 200 °C for 30 mins and 1 hour and at 250 °C for 15 minutes, 30 minutes and 1 hour to study the effect of change in temperature on the conductivity

and the stability of printed conductive features. The samples were printed with two different drop spacing values 100  $\mu\text{m}$  and 50  $\mu\text{m}$ . This chapter deals with the 100  $\mu\text{m}$  drop spacing.

The sample was sintered at different temperatures and different times in the convection oven. The electrical resistivity of the printed lines was calculated using equation (1) after sintering. The resistance of the each of the line was measured using a Fluke 117 digital multimeter. The cross-section area represented in equation as ‘Area’ of the structure was calculated as shown in equation (2).

$$\text{Resistivity} = \text{Area} \frac{\text{Resistance}}{\text{Length}} \quad (1)$$

$$\text{Area} = \text{thickness of the line} \times \text{width of the line} \quad (2)$$

$$\text{Conductivity} = \frac{1}{\text{Resistivity}} \quad (3)$$

**i) 100  $\mu\text{m}$  drop spacing, sintered at 200  $^{\circ}\text{C}$  for 30 min**

As discussed earlier, a set of six parallel lines of different lengths were printed on the glass substrate, as shown above in figure 5.1. The structure was sintered at 200  $^{\circ}\text{C}$  for 30 minutes. Figure 5.5 show the printed lines viewed in optical microscope.

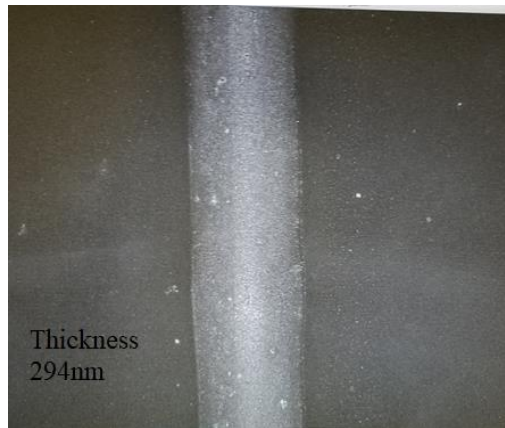


Fig. 5.5: Line printed with 100  $\mu\text{m}$  drop spacing and sintered at 200  $^{\circ}\text{C}$  for 30 min, viewed from optical microscope.

The resistance in ohm vs the length of the line in  $\mu\text{m}$  were plotted. Figure 5.6 shows the plot. The resistance is seen to increase linearly as the length of the line increases. The maximum value of  $32.6\ \Omega$  was obtained for the 60 mm line and  $10.7\ \Omega$  for the 10 mm line. The slope of the line shown in fig. ( $\Delta R$  and  $\Delta L$ ) was calculated. These values were used to calculate the resistivity in above equation (1). The value of  $\Delta R$  was assigned to the line's resistance represented as 'Resistance' in above equation (1) whereas the  $\Delta L$  value (the difference between maximum length to minimum length which is  $50,000\ \mu\text{m}$ ) was assigned to the 'Length' in above equation (1). The variation of around  $0.3\ \Omega$  is obtained which is shown as an error bar in the below fig. 5.6.

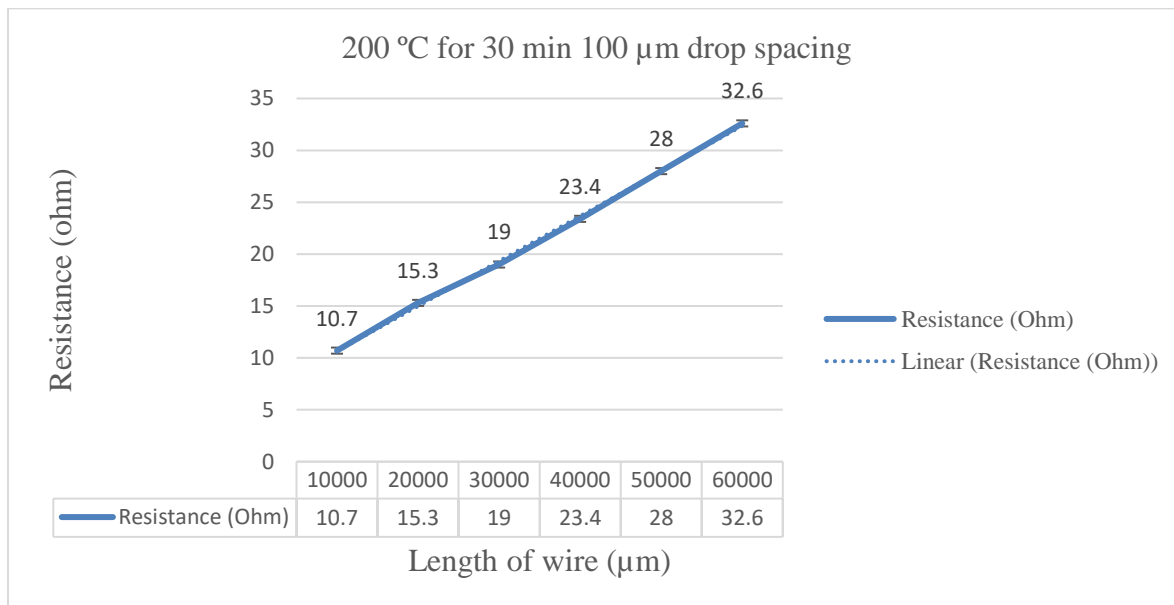


Fig. 5.6: Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for  $100\ \mu\text{m}$  drop spacing sintered at  $200\ ^\circ\text{C}$  for 30 min.

The printed pattern was analyzed using an optical microscope and an Alpha step 500 surface profiler. The Alpha step 500 surface profiler was used to calculate the thickness and the width of the printed structure. The line resistance was seen to increase linearly with the length.

The thickness of the printed structure was 294 nm and the width of the structure was 378  $\mu\text{m}$ . The thickness and the width were measured at two different locations and found to be almost same.

The calculated cross-section area of the printed line ('Area' in equation 2) is  $0.294 \mu\text{m} \times 378 \mu\text{m} = 111.132 \mu\text{m}^2$ . The conductivity was calculated as shown in equation (3). The percentage conductivity with respect to bulk silver was calculated by dividing the calculated conductivity (S/cm) with the conductivity of the bulk silver ( $6.3 \times 10^5$ ) S/cm.

The resistivity and conductivity of the printed silver lines were calculated as  $4.8 \mu\Omega\text{-cm}$  and  $2.05\text{E}+05$  S/cm respectively. This is 32.6% conductivity of bulk silver. The resistivity value for silver nanoparticles ink obtained is in the range to be expected as given in the datasheet [40]. Key performance data of the silver nanoparticles ink from the datasheet is shown in table 5.3. Table 5.4 summarizes the calculated conductivity.

Table 5.3: Key performance data of silver nanoparticles ink from the datasheet.

Curing temperature	120-300 °C
Resistivity, curing at 250 °C, 10 min	$\sim 3 \mu\Omega\text{cm}$
Resistivity, curing at 140 °C, 60 min	$\sim 10 \mu\Omega\text{cm}$
Printable Material	Polyimide, Teslin, Silicon

Table 5.4: Calculated conductivity value in S/cm at 200 °C for 30 min, drop dispensed at 150V  
30  $\mu\text{s}$  pulse width and 100  $\mu\text{m}$  drop spacing.

Temperature	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
200 °C for 30 min	$4.86 \mu\Omega\text{cm}$ $+ (6.30 \Omega \pm 0.27)$	$2.05\text{E}+05$	32.6

The resistivity value in above table also includes the offset value along with the intercept error. This offset value is the y axis intercept value obtained from fig. 5.6. The error indicates the uncertainties of the y axis intercept of the best fit line and are determined using average standard error of the y coordinates. The offset could be due to the contact resistance of the probe used for calculating the resistance using digital multimeter.

**ii) 100  $\mu\text{m}$  drop spacing, sintered at 200  $^{\circ}\text{C}$  for 60 min**

In the following experiment the sintering time was increased while keeping the same temperature. It was observed that increasing the time at this temperature does not significantly affect the conductivity value of the printed structure.

The resistance in ohm vs the length of the line in  $\mu\text{m}$  were plotted. Figure 5.7 shows the plot. The resistance is seen to increase linearly as the length of the line increases. The maximum value of 31.2  $\Omega$  was obtained for the 60 mm line and 8  $\Omega$  for the 10 mm line. The variation of around 0.6  $\Omega$  is shown as an error bars in the below fig. 5.7.

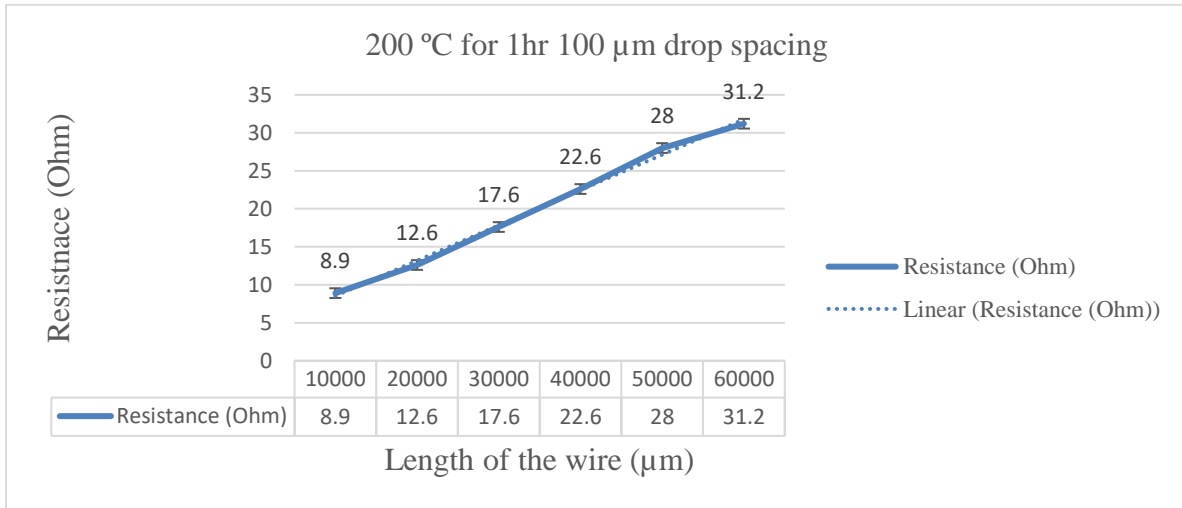


Fig. 5.7: Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for 100  $\mu\text{m}$  drop spacing sintered at 200  $^{\circ}\text{C}$  for 60 min.

The measured resistivity value remains almost the same, around 32 % conductivity of the bulk silver was obtained with a resistivity of about  $4.9 \mu\Omega\text{cm}$ . The below table 5.5 summarizes the calculated result at 200 °C for 60 min. The same procedure was followed to calculate the resistivity and conductivity of the line as done in case 1. The offset of the line was calculated in similar way as done in above case and added to the table with the offset error. The resistivity value in the table shows the same.

Table 5.5: Calculated conductivity value in S/cm at 200 °C for 60 min, drop dispensed at 150V  
30  $\mu\text{s}$  pulse width and 100  $\mu\text{m}$  drop spacing.

Temperature	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
200 °C for 60 min	$4.92 \mu\Omega\text{cm}$ $+ (3.88 \Omega \pm 0.59)$	2.03E+05	32.2

**iii) 100  $\mu\text{m}$  drop spacing, sintered at 250 °C for 15 min**

The oven temperature was increased to 250 °C for 15 minutes. It was observed that with an increase in the temperature the offset was decreasing, and the wire was becoming more and more conductive. More densifying of the material resulted in more conductive structure [16].

The resistance in ohm vs the length of the line in  $\mu\text{m}$  were plotted. Figure 5.8 shows the plot. Here also, the resistance was seen to be increased linearly as the length of the wire was increase. The maximum value of  $34.5 \Omega$  was obtained for 60 mm line and  $9.8 \Omega$  for a 10 mm line. This value was higher than that of resistance value at 200 °C form 30 min. The variation of around  $0.4 \Omega$  is shown as an error bars in the below fig. 5.8.

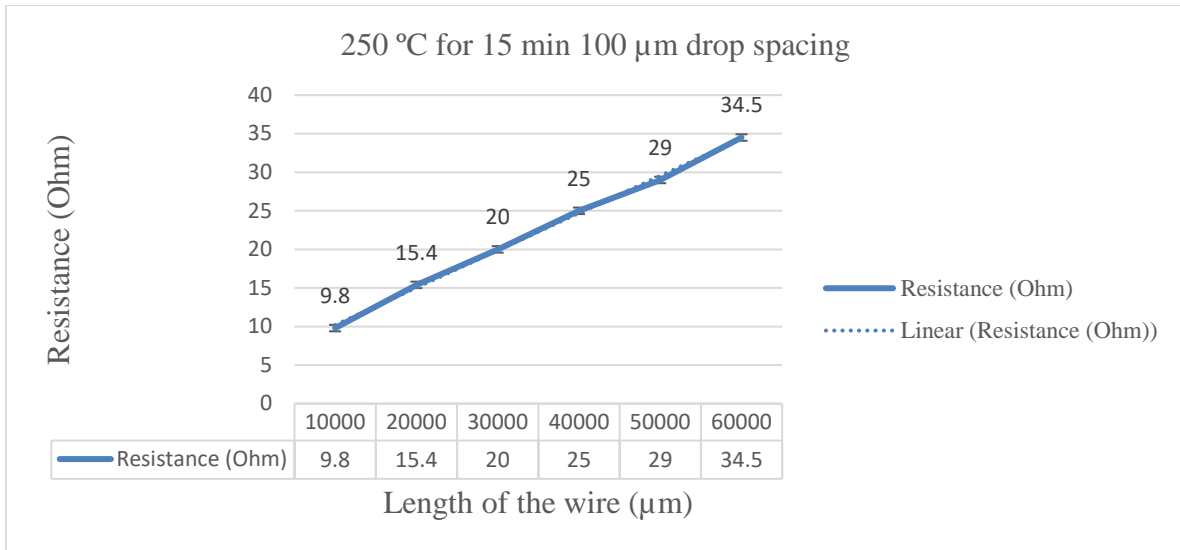


Fig. 5.8: Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for 100  $\mu\text{m}$  drop spacing sintered at 250 °C for 15 min.

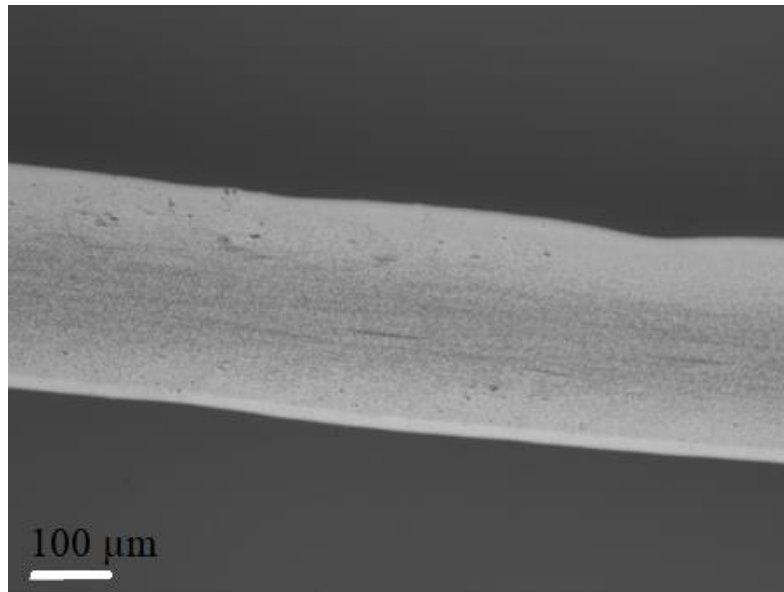


Fig. 5.9: A 100  $\mu\text{m}$  drop spacing sintered line with thickness 0.294  $\mu\text{m}$ , viewed in optical microscope.

The same procedure was followed to calculate the resistivity and conductivity of the line as done in case 1. The offset of the line was calculated in similar way as done in above case and added to the table with the offset error. The resistivity value in the table shows the same.

Table 5.6 summaries the conductivity obtained at this temperature. A 29% conductivity as compared to the conductivity of bulk silver of the printed structure was achieved at 250 °C for 15 min. The substrate was also transparent to microwave radiation in the oven. The conductive silver nanoparticles absorb the microwaves strongly resulting in uniformly heated silver tracks [16].

However, the conductivity obtained at 250 °C for 15 min was slightly less when compared with the conductivity at 200 °C for 30 min. This showed that not only the temperature, but the sintering duration always plays a vital role in the process. This has been discussed above while discussing sintering process. As discussed, sintering is a process based on diffusion, and therefore both temperature and time are essential. A resistivity of around 5.48  $\mu\Omega\text{cm}$  was obtained.

Table 5.6: Calculated conductivity value in S/cm at 250 °C for 15 min, drop dispensed at 150V  
30  $\mu\text{s}$  pulse width and 100  $\mu\text{m}$  drop spacing.

Temperature	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
250 °C for 15 min	5.48 $\mu\Omega\text{cm}$ + (5.35 $\Omega \pm 0.39$ )	1.82E+05	28.9

**iv) 100  $\mu\text{m}$  drop spacing, sintered at 250 °C for 30 min**

Now, the sintering duration time was increased to 30 min at the same temperature and the conductivity result was compared with that of the result obtained at 200 °C for 30 min.

The resistance in ohm vs the length of the line in  $\mu\text{m}$  were plotted. Figure 5.10 shows the plot. A maximum value of 24  $\Omega$  was obtained for 60 mm line and 7  $\Omega$  for a 10 mm line. The pattern was same as obtained in above cases. However, these values were less than the result obtained at



200 °C for 30 min. The variation of around 0.3  $\Omega$  in this case is shown as an error bars in the below fig. 5.10. The same procedure was followed to calculate the resistivity and conductivity of the line as done in above cases. The offset of the line was calculated in similar way as done in above case and added to the table with the offset error. The resistivity value in the table 5.7 shows the same.

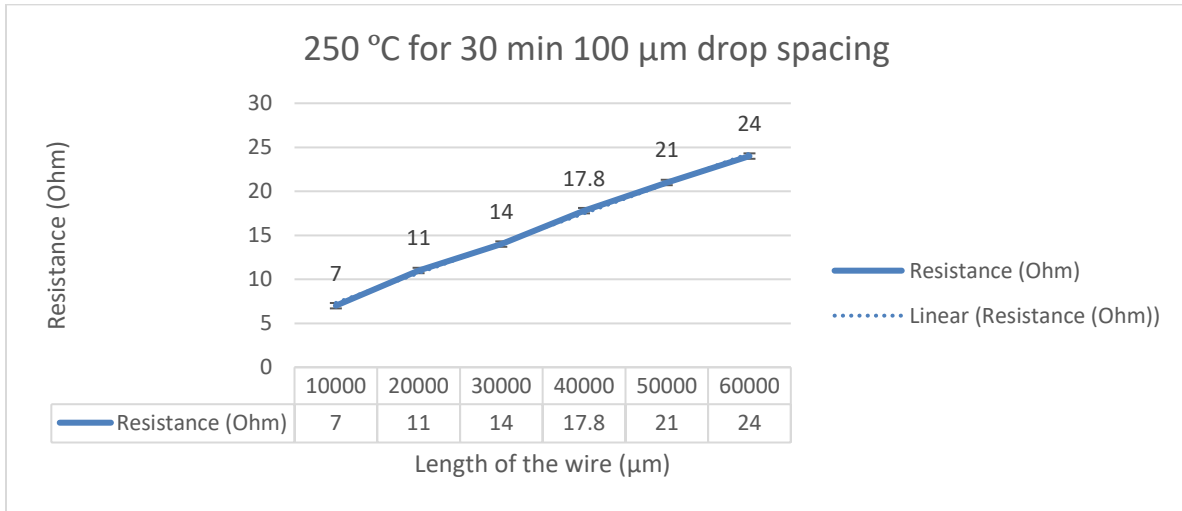


Fig. 5.10: Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for 100  $\mu\text{m}$  drop spacing sintered at 250 °C for 30 min.

Table 5.7 summaries the conductivity result obtained at 250 °C for 30 min. A very high value of 42 % conductivity as compared with the conductivity of the bulk silver, was obtained at this temperature. This was about 12 % higher than the result obtained at 200 °C for 30 min. An increase of 50 °C in temperature resulted in a good conductivity of the printed samples. A 3.8  $\mu\Omega\text{cm}$  resistivity was obtained which was less than that of the resistivity result obtained at 200 °C for 30 min.

The time was also increased to 60 min at the same temperature to study the change of the conductivity with time and it was observed that the prolonged heating at the same temperature was not affecting the conductivity of the printed structure thus suggesting that these printed structures were already subjected to sufficient sintering conditions.

Table 5.7: Calculated conductivity value in S/cm at 250 °C for 30 min, drop dispensed at 150V

30  $\mu$ s pulse width and 100  $\mu$ m drop spacing, sustains same value at 250 °C for 60 min

Temperature	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
250 °C for 30min	3.77 $\mu\Omega\text{cm}$ + (3.92 $\Omega \pm 0.28$ )	2.64E+05	42.0

It was observed that an increase in sintering temperature resulted in more conductivity of the silver particles. Figure 5.11 was plotted to show the relation between the temperature and the conductivity curve for 30 mins. It was observed that the conductivity increased to 2.64E+05 from 2.05E+05 when the temperature was increased from 200 °C to 250 °C for 30 mins for the same printed structure using similar dispensing parameters.

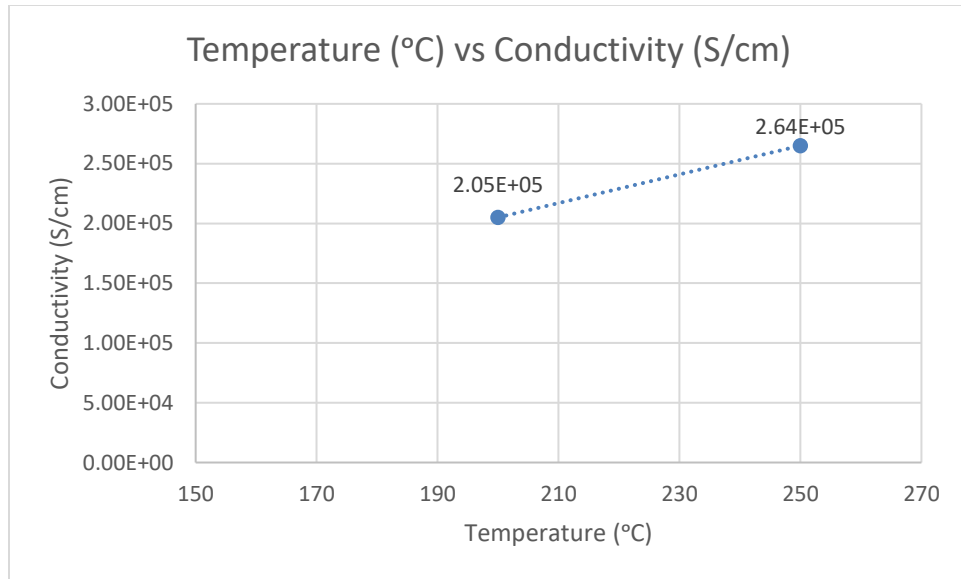


Fig. 5.11: Temperature vs Conductivity curve for 30 min at 200 °C and 250 °C.

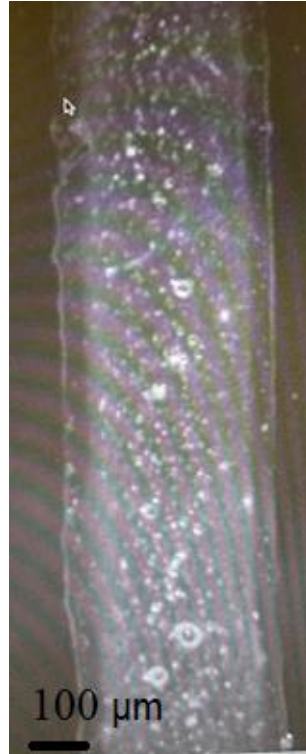


Fig. 5.12: Silver particles not sintered properly at 150°C for 1 hr, viewed in optical microscope.

### 5.3.2 An Incandescent Lamp Sintering

We have until now discussed about the oven sintering process and the conductivity result obtained at different temperature and time. We have also studied as how the change in temperature can affect the conductivity of silver nanoparticles. There is another type of sintering mechanism also known as flash sintering. This is also explored in this research work. This section deals this method. The discussion is also about the conductivity results obtained using this sintering process. Later, the obtained conductivity result is compared with the above oven sintering results.

The flexibility of the substrate is vital in printed electronic applications. It is essential that the sintering temperature must remain below the glass transition temperature of the used substrate, or, it may lead to a permanent damage to the substrate. The low-cost microscopic glass substrate used for this work suffices this need. The oven sintering process discussed above is the most

commonly used sintering method wherein the temperature range is limited. In the oven sintering, the samples were sintered for an hour to achieve a good conductivity of the printed samples. Hence, such technique is not feasible for fast industrial production because of the long sintering times.

Many alternatives approaches were used earlier to overcome this. A laser sintering method was one of them. This process was first developed and patented by Dr. Carl Deckard at the University of Texas at Austin. It was done under sponsorship of DARPA (Deckard 1989) and was later commercialized by DTM Corporation. This was subsequently bought out by 3D Systems [35][36]. The laser sintered the conductive tracks without much affecting the substrate where the printed structure is normally heated using CO<sub>2</sub> laser. However, this method is costly and complex.

This work discusses about a fast, simple, and cost-effective tungsten halogen lamp sintering technique. This technique allowed the sintering of the printed structures in a quick time. Selective heating of the printed features was done. In this technique, a tungsten halogen lamp as the light sources was used to sinter the silver particles. This light source was selected because of its high process speed and less running cost. It can also generate a continuous distribution of light across the visible spectrum.

This type of technique helps in improving the process speed for rapid manufacturing process. They are widely in synthetic chemistry [38].

#### **5.3.2.1 Experiment Results**

An intense pulsed light from 500 W 120 V tungsten halogen lamp (Hikari J-2059) was used to sinter silver particles. It was used to apply a broad-wavelength light at high energy density to increase the temperature of the printed structure on the glass substrate instantly. Here, in this case a three parallel lines of 10 mm to 30 mm were printed on a substrate. The distance between

these lines was 3mm. The dispensing parameters to dispense the ink drop were kept same as that used in oven sintering. These parameters can be viewed in table 5.2.

Figure 5.13 shows the schematic diagram describing the assembly of the printed structure on glass substrate undergoing lamp sintering process. A black color build plate is used as the substrate base as shown in figure. This was same as used in earlier cases. Two supporting glasses at 90° were used to get another inch in height. Single layer printing was done on a glass substrate kept on the black plate. The sample was moved to beneath the built structure on the Y axis of the X-Y stage once the printing is completed. The assembly of the lamp sintering process is shown in below figure. Later, the sample was moved onto the supporting glass.

One of the critical factors using an incandescent lamp sintering process is the distance between the heat source and the substrate. The intensity of energy decreases with the distance between source and the object. In this case, as the length of the energy source is long enough than the distance of the substrate to the light source. Hence, the amount of radiation passing through a specific area is inversely proportional to the distance of that area from the energy source.

Here, the distance between the heat source and the substrate were varied and the sintering process was carried out. Also, the sintering time was varied to obtain the result. Later, the result was tabulated in a summary and presented in section 5.3.3. The result included the comparison of the sintering result with the distance of the source with the substrate and the sintering time.

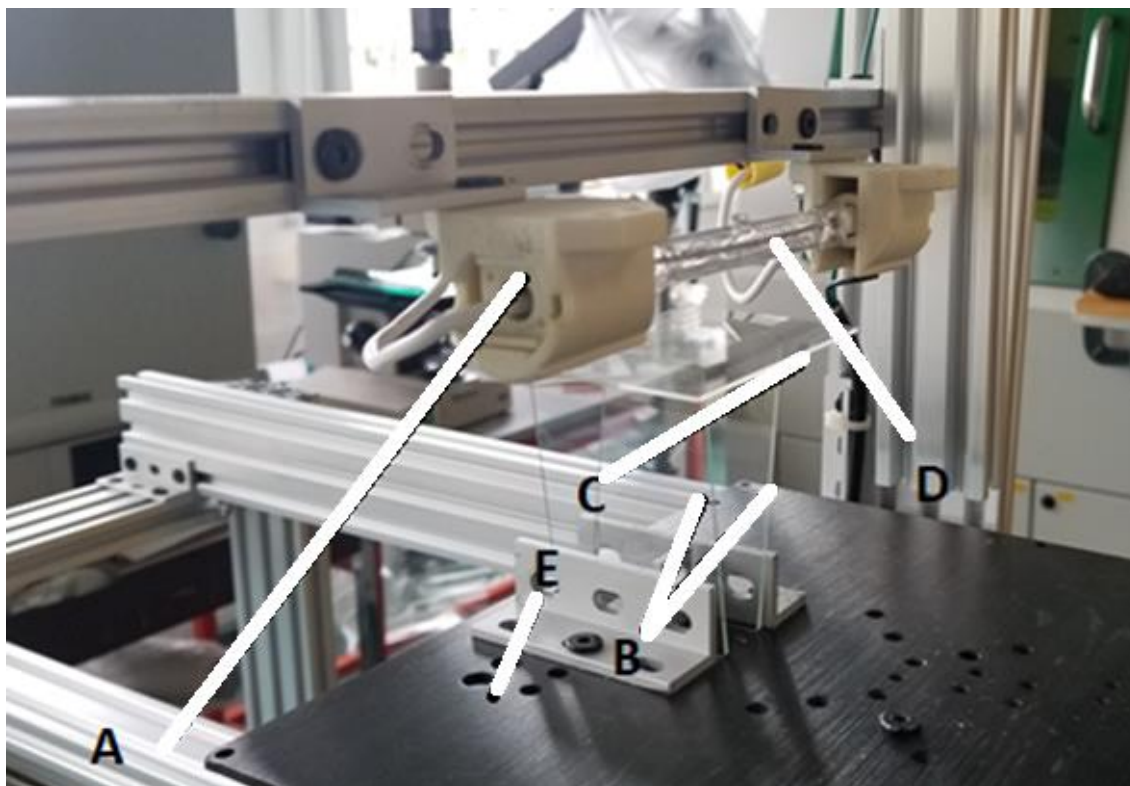


Fig. 5.13: Schematic diagram describing the assembly of the printed structure on glass substrate undergoing halogen lamp sintering. A: Bulb holder assembly. B: Two supporting glasses at 90 degrees are used to get another inch in height. C: A glass substrate. D: 500 W 120 V tungsten halogen lamp (Hikari J-2059). E: Substrate holder.

**i) Using 500-Watt Bulb, 1 cm from bulb for 2 min**

The distance between the heat source and the substrate was kept at 1 cm and the experiment was carried out for 2 minutes. The height from the bulb was measured using a metric scale ruler.

Here also, first the resistance of the three lines were calculated as done in above cases for oven sintering using the same multimeter and was plotted with the length of the line. Figure 5.14 shows the resistance in ohm vs the length of the line in  $\mu\text{m}$ . A three data points was used for carrying out this experiment work. It was observed with the increase in length, the resistance was

increased linearly. The maximum value of  $16\ \Omega$  was obtained for 30 mm line and  $5.5\ \Omega$  for a 10 mm line. The variation of around  $0.6\ \Omega$  in this case is shown as an error bars in the below fig. 5.14. The same procedure was followed to calculate the resistivity and conductivity of the line as done in case 1. The offset of the line was calculated in similar manner as done in above cases of oven sintering and added to the table 5.8. The resistivity value in the same table shows the offset value.

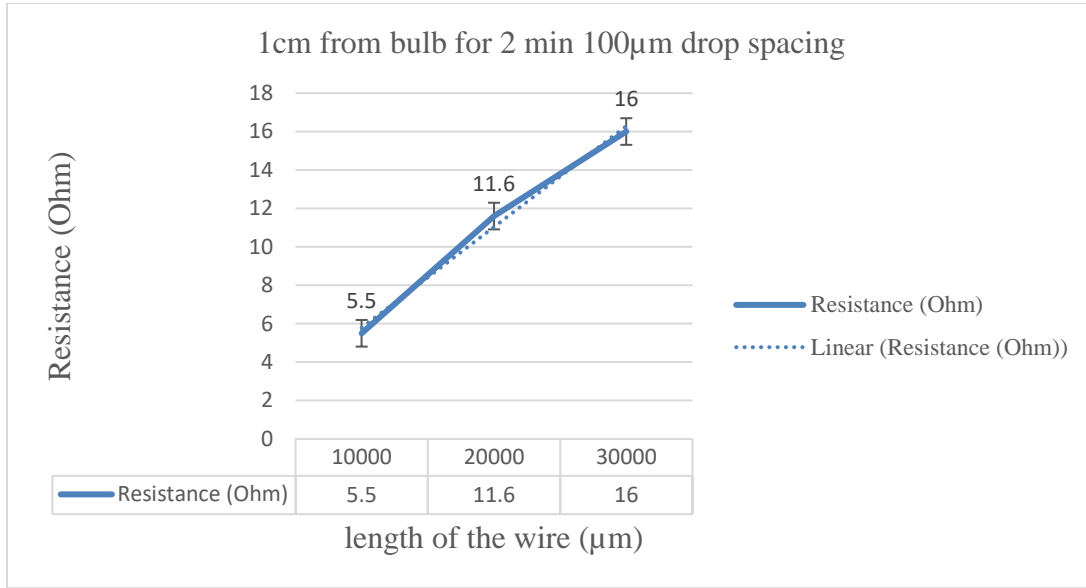


Fig. 5.14: Resistance (Ohm) vs. the length of the wire ( $\mu$ m). Resistance measured for 100  $\mu$ m drop spacing, sintered at 1 cm from bulb for 2 min.

Table 5.8 summaries the resistivity obtained of the printed structure. The resistivity and conductivity of the printed silver lines were  $5.8\ \mu\text{Ohm-cm}$  and  $1.7\text{E}+05\ \text{S/cm}$  respectively. This is about 27% conductivity of the conductivity of the bulk silver. The electrical resistivity and the conductivity of the printed lines were calculated using above equation (1) and equation (3) respectively. The offset of the line was calculated in similar way as done in above case and added to the table with the offset error. The resistivity value in the table 5.8 shows the same. The dispensing parameters for dispensing ink drop to print the structure were kept same as that discussed in above cases.

Table 5.8: Calculated conductivity value in S/cm using 500-Watt Bulb, 1 cm from bulb for 2 min, drop dispensed at 150V, 30  $\mu$ s pulse width and 100  $\mu$ m drop spacing.

Distance from the bulb and time	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
1 cm from bulb for 2 min	5.83 $\mu\Omega\text{cm}$ + (0.53 $\Omega \pm 1.06$ )	1.71E+05	27.2

It was observed that less than 2 minutes at 1 cm from bulb, the ink was not conductive. The time was decreased to 1.5 minutes and 1 minute at 1 cm from bulb. Decreasing the energy density either by decreasing the time or by increasing the distance was not providing the conductivity result.

Increasing the time more than 2 minutes to 2.2 minutes resulted in the breaking of the glass substrate. The distance was also decreased to 0.5 cm and the sintering was carried out for 50 seconds, but the patterns were not yielding the conductivity result. This shows that, as discussed earlier in section 5.2, that sintering process not only depends on the heat, but the time also plays an important role. It is a diffusion process.

### 5.3.3 Sintering Summary

This section deals with comparing the conductivity and resistivity results at different temperature and time using oven sintering method. It also includes the comparison of the sintering results obtained using incandescent lamp sintering method. Later, the results were compared using both these two sintering methods. Table 5.9 and 5.10 shows the sintering summary results.



Table 5.9: Sintering using oven method at different temperature and time, drop dispensed at 150V, 30  $\mu$ s pulse width and 100  $\mu$ m drop spacing.

Sintering Parameters	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
200 °C for 30 min	4.86 $\mu\Omega\text{cm}$ + (6.30 $\Omega \pm 0.27$ )	2.05E+05	32.6
200 °C for 60 min	4.92 $\mu\Omega\text{cm}$ + (3.88 $\Omega \pm 0.59$ )	2.03E+05	32.2
250 °C for 15 min	5.48 $\mu\Omega\text{cm}$ + (5.35 $\Omega \pm 0.39$ )	1.82E+05	28.9
250 °C for 30 min	3.77 $\mu\Omega\text{cm}$ + (3.92 $\Omega \pm 0.28$ )	2.64E+05	42.0
250 °C for 60 min	3.84 $\mu\Omega\text{cm}$ + (3.5 $\Omega \pm 0.46$ )	2.63E+05	41.7

Table 5.10: Sintering using incandescent lamp and varying the printed structure distance from the bulb and time, drop dispensed at 150V, 30  $\mu$ s pulse width and 100  $\mu$ m drop spacing.

Distance from the bulb and time	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
1 cm from bulb for 1 min	-	No conductivity	-
1 cm from bulb for 1.5 min	-	No conductivity	-
1 cm from bulb for 2 min	5.83 $\mu\Omega\text{cm}$ + (0.53 $\Omega \pm 1.06$ )	1.71E+05 S/cm	27.2
1 cm from bulb for 2.2 min	-	Glass substrate breaks.	-
0.5 cm from bulb for 50 seconds	-	No conductivity	-
0.5 cm from bulb for 1min	-	Glass substrate breaks.	-

The result obtained using incandescent lamp sintering process was about 2% less conductivity of bulk silver as compared with convection oven sintering process at 250 °C for 15 mins. The high resistance of the sintered silver ink pattern produced by tungsten halogen lamp was because of incomplete sintering of the silver nano ink pattern through the thickness. A  $1.7140\text{E}+05$  S/cm conductivity was obtained just in 2 min as compared to  $1.8215\text{E}+05$  S/cm conductivity when sintered in a convection oven at 250 °C for 15 minutes and hence was very fast and cost-effective.

The silver track printed on a glass substrate showed no conductivity when sintered in oven at 150 °C for 1 hour. The printed track could be easily scratched when rubbed with fingers under such condition suggesting a bad adhesion when not properly sintered.

The silver track printed on a glass substrate started showing conductivity when sintered in oven at temperature 200 °C and greater. The silver track couldn't be scratched implying better adhesion with glass substrate when sintered properly.

## **Chapter 6: Change Dispensing Parameters to Study the Effect of line width**

### **6.1 Introduction**

We have discussed about creating patterns with silver nanoparticles without changing the drop spacing and dispensing parameters in previous chapter. The ink drops of silver nanoparticles were dispensed with a 100  $\mu\text{m}$  drop spacing and at 150 V and 30  $\mu\text{s}$  pulse width for printing the patterns. We have also discussed about two types of sintering process. Here, in this chapter the discussion is mainly about creating patterns with the same silver ink and varying the drop spacing and the dispensing parameters such as voltage and the pulse width and study the change in the thickness of the patterns. The patterns were printed with a 50  $\mu\text{m}$  drop spacing. The printed pattern was oven sintered at 250  $^{\circ}\text{C}$  for 30 minutes.

### **6.2 Dot-to-Dot Spacing**

The dot spacing affects the width and the thickness of the printed structure during printing. The drop spacing is the linear distance between drops during printing. It was observed that decreasing the distance between adjacent drops increases the volume deposited per unit length. It resulted in spreading the printed line to a wider width. The resolution was decreased. An increase in dot-to-dot spacing resulted in decreasing the line width.

The drop spacing was changed from the user interface page and the six parallel lines of length from 10 mm to 60 mm with 1 $\times$ 1 mm rectangular pad at both ends of the line were printed on a glass substrate. The similar structure was printed in last chapter. The printed structure was oven sintered.

The dispensing parameters were kept constant and the drop spacing was reduced from 100  $\mu\text{m}$  to 50  $\mu\text{m}$ . The dispensing parameters to dispense the ink drop can be viewed in table 5.2 of

chapter 5. It was observed that the thickness of the printed structure was increased from 294 nm to 355 nm (This value was an average of 356 nm, 356 nm and 353 nm) (cross profile of printed silver track as viewed in Alpha step surface profiler is shown in fig.6.2.) This was about an increase of 21%. The width of the printed line increased from 378  $\mu\text{m}$  to 619  $\mu\text{m}$  (This value was an average of 620  $\mu\text{m}$ , 618  $\mu\text{m}$  and 618 $\mu\text{m}$  and changed to three significant figures). This is 63 % increase. Thus, reducing the drop spacing to half increases the thickness and width of the printed feature to 21 % and 63 % respectively.



Fig. 6.1: A line drawn with 50  $\mu\text{m}$  drop spacing, sintered at 250  $^{\circ}\text{C}$  for 30 min, viewed in optical microscope. Calculated line thickness 355 nm and line width 619  $\mu\text{m}$ . Clearly indicating increase in line thickness and line width.

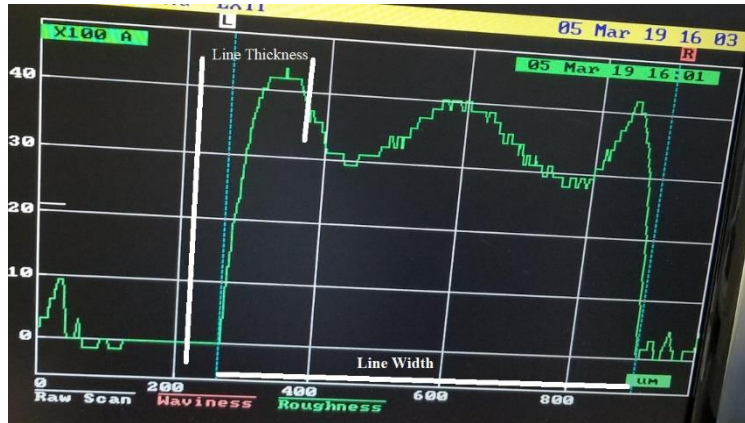


Fig. 6.2: Cross profile of printed silver track as viewed in Alpha step 500 surface profiler.

The silver track was printed with 50  $\mu\text{m}$  drop spacing with a line thickness of about 355 nm and line width of 619  $\mu\text{m}$ , sintered at 250  $^{\circ}\text{C}$  for 30 min.

### 6.3 Experiment Results

Here, the dispensing parameters were changed, and the same parallel lines of six different lengths varying from 10 mm to 60 mm were printed on the glass substrate. This printed structure is same as done in previous chapter 5.

The voltage was changed at same pulse width and the results were obtained. Also, the pulse width was changed keeping voltage value constant. The printed structure was oven sintered at 250  $^{\circ}\text{C}$  for 30 min. Patterns were printed at room temperature. The nozzle and substrate temperature were kept at room temperature.

1. **No change in dispensing parameters which means drop dispensed at 150 V and 30  $\mu\text{s}$  width. Samples printed with 50  $\mu\text{m}$  drop spacing and oven sintered at 250  $^{\circ}\text{C}$  for 30 min.**

The dispensing parameters were kept constant as described in table 5.2 in chapter 5 and only the drop spacing value was changed from 100  $\mu\text{m}$  to 50  $\mu\text{m}$ . Table 6.2 summaries the conductivity value in S/cm obtained after sintering at 250  $^{\circ}\text{C}$  for 30 min.

It was observed that change in the drop spacing from 100  $\mu\text{m}$  to 50  $\mu\text{m}$  increases the thickness and width of the printed line and hence, the area of the printed structure. Here also, the resistance of the lines was calculated first as done in all above cases for oven sintering in previous chapter 5 using the same multimeter and was plotted with the length of the lines. The below fig. 6.3 shows the resistance in ohm vs the length of the line in  $\mu\text{m}$ .

It was observed that the resistance was linearly increased with increase in the length of the wire. The maximum value of 17  $\Omega$  was obtained for 60 mm line and 3.5  $\Omega$  for a 10 mm line. The resistance obtained was less than the value obtained for structure drawn with 100  $\mu\text{m}$  drop spacing for each of the length. A variation of around 0.3  $\Omega$  in this case is shown as an error bars in the below figure. The offset of the line was calculated similarly as done in previous chapter 5 and added to the table 6.2. The resistivity value in the same table shows the offset value with the offset error. The offset arises due to the reason as stated in previous chapter 5.

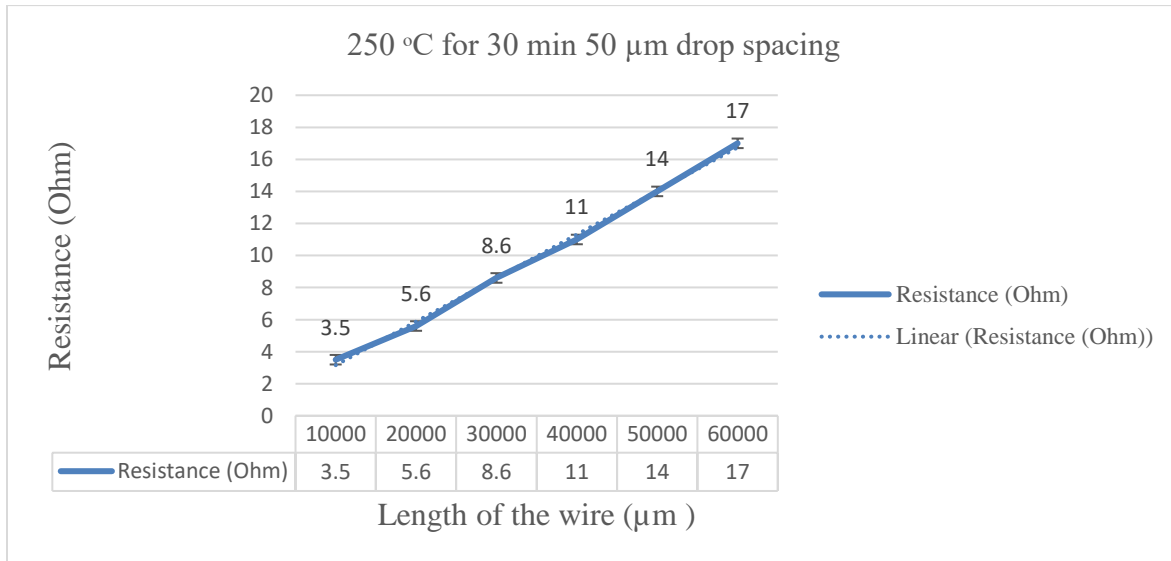


Fig. 6.3: Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for 50  $\mu\text{m}$  drop spacing sintered at 250 °C for 30 min.

The surface profiler was used to calculate the thickness and the width of the printed line. The thickness of the printed line for a 50  $\mu\text{m}$  drop spacing was 355 nm (This value was an average of 356 nm, 356 nm and 353 nm and changed to three significant numbers). The width of the printed line for a 50  $\mu\text{m}$  drop spacing was 619  $\mu\text{m}$  (This value was an average of 620  $\mu\text{m}$ , 618  $\mu\text{m}$  and 618  $\mu\text{m}$  and changed to three significant numbers). Error indicates extreme range of the variation in the line thickness and the line width. The value was added the table as listed below. The same was done for all the cases in this chapter. The increased cross section area of the printed structure was calculated using equation (2) as described in chapter 5. The calculated area of the line ( $0.355 \mu\text{m} \times 618.6 \mu\text{m} = 219.745 \mu\text{m}^2$ ). This was clearly more than the cross-section area of the printed structure with 100  $\mu\text{m}$  drop spacing which was  $111.132 \mu\text{m}^2$ . The increase in the cross-section area of the printed structure resulted in the increase resistivity value.

Table 6.1: Comparison of the change in the line width and the thickness of the printed structure at different drop spacing for same dispensing parameters.

Dispensing Parameters	Drop spacing	Line thickness $\pm$ error	Line width $\pm$ error
150 V, 30 $\mu\text{s}$ pulse width	100 $\mu\text{m}$	(294 $\pm$ 15) nm	(378 $\pm$ 10) $\mu\text{m}$
150 V, 30 $\mu\text{s}$ pulse width	50 $\mu\text{m}$	(355 $\pm$ 15) nm	(619 $\pm$ 10) $\mu\text{m}$

The conductivity of the line was calculated using equation (3) described in chapter 5. The percentage conductivity with respect to bulk silver was done in similar manner as done in previous chapter 5. A  $5.93 \mu\Omega\text{cm}$  resistivity was obtained. This was about 56 % more of the value obtained for the same structure drawn with 100  $\mu\text{m}$  drop spacing ( $3.8 \mu\Omega\text{cm}$ ).

The conductivity obtained was  $1.68\text{E}+05 \text{ S/cm}$ . This was about 26 % conductivity as compared to the conductivity of the bulk silver. A 42 % conductivity of bulk silver was obtained

for structure drawn with 100  $\mu\text{m}$  drop spacing and oven sintered at 250  $^{\circ}\text{C}$  for 30 min. This shows a decrease of about 16 %.

Table 6.2: Calculated conductivity value in S/cm at 250  $^{\circ}\text{C}$  for 30 min for structure printed with 50 $\mu\text{m}$  drop spacing at 150 V and 30  $\mu\text{s}$  pulse width

Temperature	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
250 $^{\circ}\text{C}$ for 30 min	5.93 $\mu\Omega\text{cm}$ + (0.44 $\Omega \pm 0.27$ )	1.68E+05	26.7

**2. No change in dispensing parameters which means drop dispensed at 150 V and 30  $\mu\text{s}$  width. Samples printed with 50  $\mu\text{m}$  drop spacing and oven sintered at 250  $^{\circ}\text{C}$  for 60 min.**

Here in this case, only the time duration was increased at the same 250  $^{\circ}\text{C}$  to study the change in the value of conductivity with time. The ink drop was dispensed using the same parameters as done in the previous case.

Here also, the resistance of the lines was calculated first as done in all previous cases using the same multimeter and were plotted with the length of the lines. Figure 6.3 shows the resistance in ohm vs the length of the line in  $\mu\text{m}$ . The printed lines were sintered in oven at 250  $^{\circ}\text{C}$  for 60 min. The maximum value of 13.5  $\Omega$  was obtained for 60 mm line and 3  $\Omega$  for a 10 mm line. The printed structure was becoming more conductive. The variation of around 0.2  $\Omega$  in this case is shown as an error bars in the below fig. 6.4.



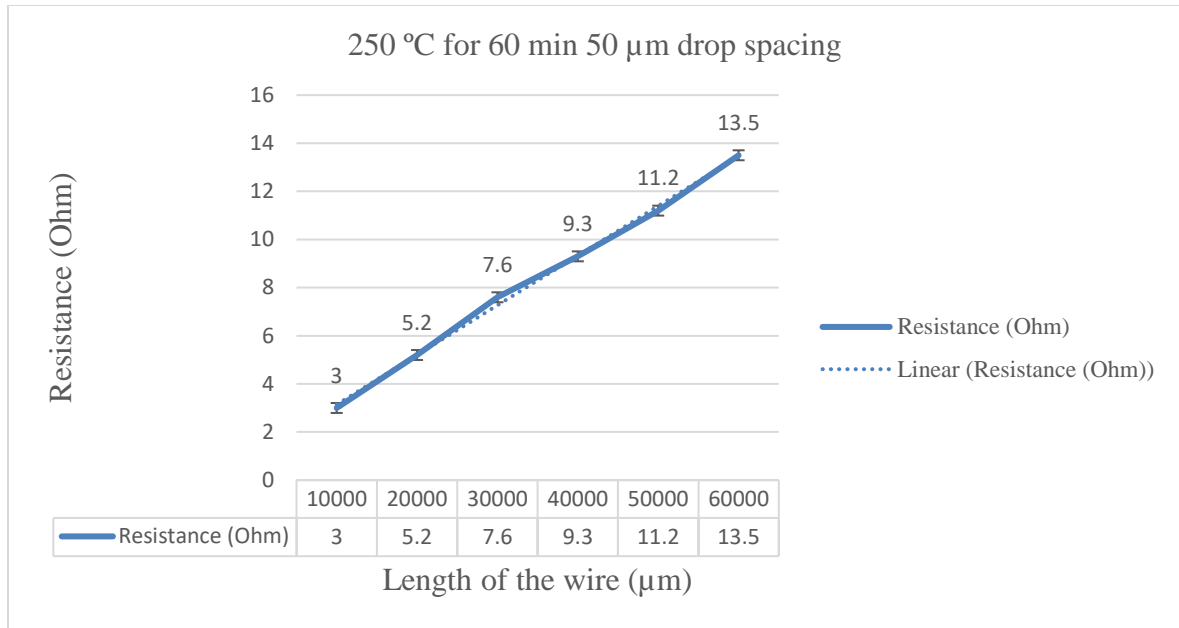


Fig. 6.4: Resistance (Ohm) vs the length of the wire ( $\mu\text{m}$ ). Resistance measured for 50  $\mu\text{m}$  drop spacing at 150 V and 20  $\mu\text{s}$ , sintered at 250 °C for 60 min.

The same procedure was followed to calculate the resistivity and conductivity of the line as done in all previous cases. The conductivity and the percentage conductivity were calculated similarly as done in previous chapter 5. A 4.6  $\mu\Omega\text{cm}$  resistivity was obtained. The conductivity obtained was 2.16E+05 S/cm. This was about 34 % of conductivity of bulk silver which was increased from 26 % of conductivity of bulk silver for structure printed when sintered for 30 min at same temperature. The offset of the line was calculated in similar way as done in above case and added to the table with the offset error. The resistivity value in the below table 6.3 shows the same.

Table 6.3: Calculated conductivity value in S/cm at 250 °C for 60 min for structure printed with 50 $\mu\text{m}$  drop spacing at 150 V and 30  $\mu\text{s}$

Temperature	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
250 °C for 60 min	4.61 $\mu\Omega\text{cm}$ + (1.08 $\Omega \pm 0.19$ )	2.16E+05	34.4

**3. The pulse width was changed from 30  $\mu\text{s}$  to 20  $\mu\text{s}$  at 150 V and samples were printed with 50  $\mu\text{m}$  drop spacing and sintered at 250  $^{\circ}\text{C}$  for 30 min.**

Here in this case, only the pulse width was changed from 30  $\mu\text{s}$  to 20  $\mu\text{s}$  keeping all other parameters same as that of first case. The ink drop was not dispensed at pulse width less than 20 $\mu\text{s}$  and more than 30  $\mu\text{s}$  at same voltage level. It was observed that decreasing the pulse width from 30  $\mu\text{s}$  to 20  $\mu\text{s}$  at same 150 V voltage with drop spacing of 50  $\mu\text{m}$  slightly decreases the thickness of the printed line. Hence, the area of the printed line.

Here also, the resistance of the lines was calculated first as done in previous cases for oven sintering in chapter 5 using the same multimeter and was plotted with the length of the lines. Figure 6.5 shows the resistance in ohm vs the length of the line in  $\mu\text{m}$ . The lines were sintered in oven at 250  $^{\circ}\text{C}$  for 30 min. The maximum value of 20  $\Omega$  was obtained for 60 mm line and a 4  $\Omega$  for a 10 mm line. The resistance obtained for the structure printed at 150 V and 20  $\mu\text{s}$  was more than the value obtained for structure printed at 150 V and 30  $\mu\text{s}$  for each of the length. A variation of around 0.3  $\Omega$  in this case is shown as an error bars in the below figure. The offset of the line was calculated similarly as done in previous chapter 5 and added to the table 6.5. The resistivity value in the same table shows the offset value.

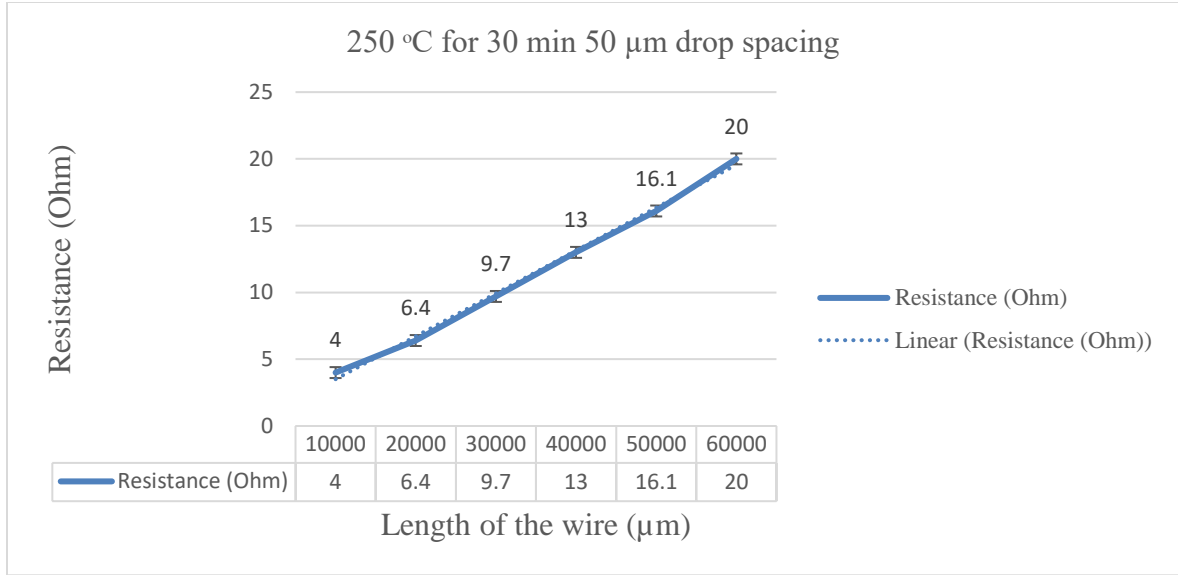


Fig. 6.5: Resistance (Ohm) vs the length of the wire (μm). Resistance measured for 50 μm drop spacing at 150 V and 20 μs, sintered at 250 °C for 30 min.

The thickness of the printed line with 50 μm drop spacing at 150 V and 20 μs pulse width was 308 nm (This value was an average of 307 nm, 307 nm, 310 nm). The width of the printed line with a 50 μm drop spacing was 525 μm (This value is an average of 528 μm, 524 μm and 524 μm and changed to three significant numbers). The pulse width was decreased from 30 μs to 20 μs at same 150 V voltage level. This resulted in decreasing the thickness of printed line from 355 nm to 308 nm. This is about 13% decrease. The width of the printed structure was decreased from 619 μm to 525 μm. This is about 15% decrease. The decreased cross section area of the printed structure was calculated using equation (2) as described in chapter 5 which comes out to be  $(0.308 \mu\text{m} \times 525 \mu\text{m} = 161.7 \mu\text{m}^2)$ . This was clearly less than the cross-section area of the printed structure at 30 μs pulse width.

Table 6.4: Comparison of the change in the line width and the thickness of the printed structure at decreased pulse width and same drop spacing.

Dispensing Parameters	Drop spacing	Line thickness $\pm$ error	Line width $\pm$ error
150 V, 30 $\mu$ s pulse width	50 $\mu$ m	(355 $\pm$ 15) nm	(619 $\pm$ 10) $\mu$ m
150 V, 20 $\mu$ s pulse width	50 $\mu$ m	(308 $\pm$ 15) nm	(525 $\pm$ 10) $\mu$ m

The same procedure was followed to calculate the resistivity and conductivity of the line as done in previous chapter. The conductivity and the percentage conductivity were calculated in similar as done in previous chapter. A 5.17  $\mu\Omega$ cm resistivity was obtained. A slight decrease in value was observed as compared to that of structure printed with 50  $\mu$ m drop spacing at 30  $\mu$ s and 150 V (5.93  $\mu\Omega$ cm).

The conductivity obtained was 1.93E+05 S/cm. This was about 30 % of conductivity of bulk silver. We have seen before, that a 26 % conductivity of bulk silver was obtained for structure printed at 30  $\mu$ s. Clearly, decreasing pulse width, keeping all other dispensing parameters same resulted in an increase of about 4 % of the conductivity of bulk silver.

Table 6.5: Calculated conductivity value in S/cm at 250  $^{\circ}$ C for 30 min for structure printed with 50 $\mu$ m drop spacing at 150 V and 20  $\mu$ s

Temperature	Resistivity ( $\mu\Omega$ cm) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
250 $^{\circ}$ C for 30 min	5.17 $\mu\Omega$ cm + (0.29 $\Omega \pm$ 0.38)	1.93E+05	30.6

**4. The voltage was changed from 150 V to 170 V and 30  $\mu$ s pulse width. Samples were printed with 50  $\mu$ m drop spacing and sintered at 250 C for 30 min.**

The same structure was printed changing the voltage level from 150 V to 170 V at same 30 $\mu$ s pulse width. Table 6.6 summaries the conductivity value in S/cm obtained after sintering at 250 °C for 30 min for this case. It was observed that increasing the voltage level at same the pulse width slightly increasing the thickness and thus the cross-section area of the printed line.

Here also, the resistance of the lines was calculated first as done in previous cases for oven sintering in chapter 5 using the same multimeter and was plotted with the length of the lines. The below fig. 6.6 shows the resistance in ohm vs the length of the line in  $\mu$ m. The lines were sintered in oven at 250 °C for 30 min. The maximum value of 15 Ohm was obtained for 60 mm line and a 3.1 Ohm for a 10 mm line. The resistance obtained for the structure printed at 170 V and 30  $\mu$ s was slightly less than the value obtained for structure printed at 150 V and 30  $\mu$ s. A variation of around 0.3  $\Omega$  is obtained which is shown as an error bars in the below figure. The offset of the line was calculated similarly as done in previous chapter 5 and added to the table 6.7. The resistivity value in the same table shows the offset value.

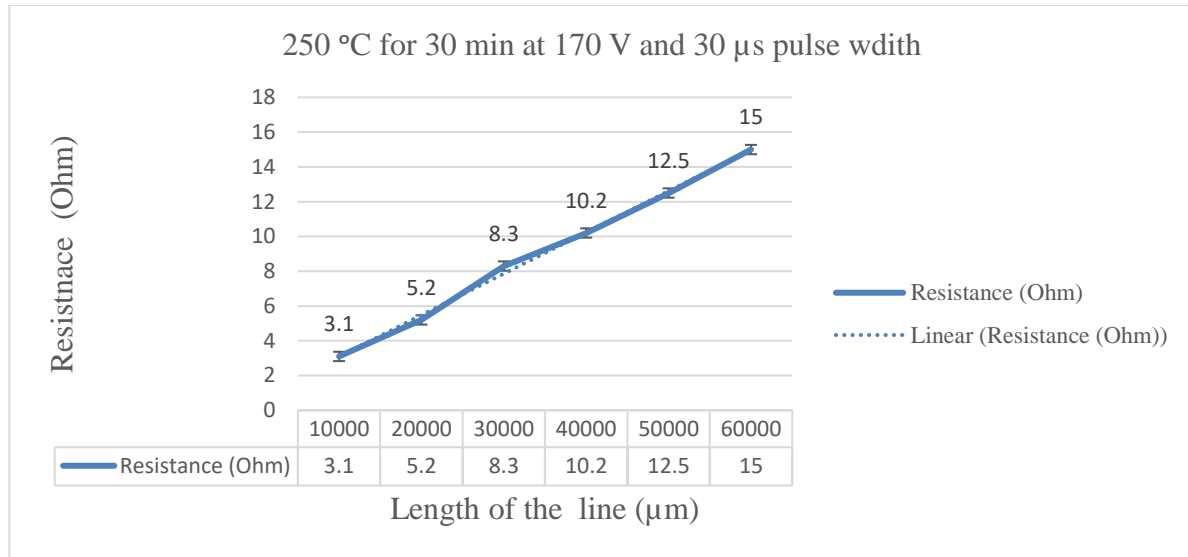


Fig. 6.6: Resistance (Ohm) vs the length of the wire ( $\mu$ m). Resistance measured for 50  $\mu$ m drop spacing at 170 V and 30  $\mu$ s, sintered at 250 °C for 30 min.

The thickness of the printed line with a 50  $\mu$ m drop spacing at 170 V and 30  $\mu$ s pulse width was 565 nm (This value is an average of 570 nm, 558 nm, 568 nm and changed to three significant numbers) and calculated width of the printed line was 583  $\mu$ m (This value is an average of 580  $\mu$ m, 580  $\mu$ m and 590  $\mu$ m and changed to three significant numbers). Increasing the voltage level at same pulse width resulted in increase in the thickness of the printed structure from 355 nm to 565 nm.

Table 6.6: Comparison of the change in the line width and the thickness of the printed structure at increased voltage level keeping same pulse width and drop spacing.

Dispensing Parameters	Drop spacing	Line thickness $\pm$ error	Line width $\pm$ error
150 V, 30 $\mu$ s pulse width	50 $\mu$ m	(355 $\pm$ 15) nm	(619 $\pm$ 10) $\mu$ m
170 V, 30 $\mu$ s pulse width	50 $\mu$ m	(565 $\pm$ 15) nm	(583 $\pm$ 10) $\mu$ m

The cross-section area of the printed structure was calculated using equation (2) as described in last chapter. This value comes out to be ( $0.565 \mu\text{m} \times 583 \mu\text{m} = 329.395 \mu\text{m}^2$ ). This was clearly more than the cross-section area of the printed structure at 150 V and 30  $\mu\text{s}$  pulse width ( $219.745 \mu\text{m}^2$ ).

The same procedure was followed to calculate the resistivity and conductivity of the line as done in previous chapter 5. The conductivity and the percentage conductivity of the bulk silver of the printed line were calculated similarly as described in earlier cases. The increased cross-section area of the printed structure resulted in increase in resistivity value. The conductivity of the printed line was decreased.

The resistivity calculated was around 7  $\mu\Omega\text{cm}$ . This is an increase of about 18 % of the value obtained for the structure printed with 50  $\mu\text{m}$  drop spacing at 30  $\mu\text{s}$  and 150 V (5.93  $\mu\Omega\text{cm}$ ).

The conductivity was around 1.27E+05 S/cm. This is about 20 % of conductivity of bulk silver. A 26 % conductivity of bulk silver was obtained for structure printed with 50  $\mu\text{m}$  drop spacing at 30  $\mu\text{s}$  and 150 V and oven sintered at 250 °C for 30 min. This was a slight decrease in the conductivity of bulk silver as the printed lines were becoming thicker.

Table 6.7: Calculated conductivity value in S/cm at 250 °C for 30 min for structure printed with 50 $\mu\text{m}$  drop spacing at 170 V and 30  $\mu\text{s}$

Temperature	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
250 °C for 30 min	7.83 $\mu\Omega\text{cm}$ + (0.72 $\Omega \pm 0.24$ )	1.27E+05	20.2

## 6.4 Sintering Summary

This section deals with summary of the results obtained by varying the dispensing parameters such as voltage, pulse width. Drop was dispensed with 50  $\mu\text{m}$  drop spacing and pattern was sintered at 250  $^{\circ}\text{C}$  for 30 min and 60 min using oven sintering method. Table 6.8 shows the result. In previous chapter 5, drop was dispensed with 100  $\mu\text{m}$  drop spacing.

Table 6.8: Comparison of sintering result when drop dispensed at different dispensing parameters and with 50  $\mu\text{m}$  drop spacing. Pattern sintered at 250  $^{\circ}\text{C}$  for 30 min using oven sintering method.

Dispensing Parameters	Drop spacing	Sintering Parameters	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
150 V, 30 $\mu\text{s}$ pulse width	50 $\mu\text{m}$	250 $^{\circ}\text{C}$ for 30 min	5.93 $\mu\Omega\text{cm}$ + (0.44 $\Omega \pm 0.27$ )	1.68E+05	26.7
150 V, 20 $\mu\text{s}$ pulse width	50 $\mu\text{m}$	250 $^{\circ}\text{C}$ for 30 min	5.17 $\mu\Omega\text{cm}$ + (0.29 $\Omega \pm 0.38$ )	1.93E+05	30.6
170 V, 30 $\mu\text{s}$ pulse width	50 $\mu\text{m}$	250 $^{\circ}\text{C}$ for 30 min	7.83 $\mu\Omega\text{cm}$ + (0.72 $\Omega \pm 0.24$ )	1.27E+05	20.2

Table 6.9: Comparison of line thickness and line width at different dispensing parameters

Dispensing Parameters	Drop Spacing	Line Thickness $\pm$ error	Line Width $\pm$ error	Cross Section Area of Printed Structure
150 V, 30 $\mu\text{s}$ pulse width	50 $\mu\text{m}$	(355 $\pm$ 15) nm	(619 $\pm$ 10) $\mu\text{m}$	219.745 $\mu\text{m}^2$
150 V, 20 $\mu\text{s}$ pulse width	50 $\mu\text{m}$	(308 $\pm$ 15) nm	(525 $\pm$ 10) $\mu\text{m}$	161.7 $\mu\text{m}^2$
170 V, 30 $\mu\text{s}$ pulse width	50 $\mu\text{m}$	(565 $\pm$ 15) nm	(583 $\pm$ 10) $\mu\text{m}$	329.395 $\mu\text{m}^2$



Table 6.10: Comparison of the sintering result when drop dispensed at 150V, 30  $\mu$ s pulse width and with two different drops spacing. Pattern sintered at 250 °C for 30 min and 60 min using oven sintering.

Dispensing Parameters	Drop spacing	Sintering Parameters	Resistivity ( $\mu\Omega\text{cm}$ ) + (Offset $\pm$ error)	Conductivity (S/cm)	% Conductivity of Bulk Silver
[from chapter 5] 150 V, 30 $\mu$ s pulse width	100 $\mu\text{m}$	250 °C for 30 min	3.77 $\mu\Omega\text{cm}$ + (3.92 $\Omega \pm 0.28$ )	2.64E+05	42.0
[from chapter 5] 150 V, 30 $\mu$ s pulse width	100 $\mu\text{m}$	250 °C for 60 min	3.84 $\mu\Omega\text{cm}$ + (3.43 $\Omega \pm 0.43$ )	2.60E+05	41.2
150 V, 30 $\mu$ s pulse width	50 $\mu\text{m}$	250 °C for 30 min	5.93 $\mu\Omega\text{cm}$ + (0.44 $\Omega \pm 0.27$ )	1.68E+05	26.7
150 V, 30 $\mu$ s pulse width	50 $\mu\text{m}$	250 °C for 60 min	4.61 $\mu\Omega\text{cm}$ + (1.08 $\Omega \pm 0.19$ )	2.16E+05	34.4

## Chapter 7: Conclusions and Future work

### 7.1 Conclusions

Inkjet printing was carried out using a micro dispensing system from Microdrop Technologies with a 70  $\mu\text{m}$  nozzle diameter MD-K-140 dispenser head. A UTDAgIJ conductive silver nano inks of average particle size of around 10 nm and a viscosity of around 11 cP was dispensed. This was dispensed to fabricate a single layer conductive lines on a commercially available 1mm thick glass substrate. The printing was carried out at room temperature.

Additionally, pre and post inkjet processing have also been studied. The glass substrates were thoroughly cleaned by washing subsequently with DI water, acetone and isopropanol and kept for few hours before printing patterns. Post-inkjet processing is also known as sintering process. This process was done to form the connections between neighbouring molecules and form the highly conductive features. Oven sintering and an incandescent lamp sintering were used.

The effect of conductivity with sintering temperature and time were studied. It was observed that an increase in sintering temperature results in more conductivity of the silver particles. A maximum conductivity of  $2.6466\text{E}+05$  S/cm at 250 °C for 30 min was obtained using oven sintering. This was about 42% of the conductivity of bulk silver. A very low  $3.8\ \mu\Omega\text{cm}$  resistivity was obtained at 250 °C for 30 min.

Later, oven sintering results were compared with incandescent lamp sintering. A 500 W 120 V tungsten halogen lamp was used for this purpose. It was observed that  $1.7140\text{E}+05$  S/cm conductivity of the printed structure was obtained just in 2 min. The substrate was kept at 1 cm from the lamp. This was about 27% conductivity of bulk silver. This conductivity value is almost similar, slightly less as compared to conductive value ( $1.8215\text{E}+05$  S/cm) obtained using oven

sintering at 250 °C for 15 minutes. It was also observed that decreasing the time at same distance was not sintering properly. Also, decreasing the distance to about half a cm was also not effective as the energy generated by the 500 W lamp was breaking the glass substrate in about 50 seconds. It was observed that sintering is a process based on diffusion, in which both temperature and time plays an important part.

A X, Y stage was built, and a system architecture was designed in java to communicate with motor and the dispenser controller. The motor controller executes the motor movement while the dispensing controller control the dispensing parameters required to generate the ink drops. Application program was designed to create a pattern. An interactive user interface was designed to create a pattern, parse it and transmit the command to interact with these modules. User were provided an option to select the pattern with start and end coordinates along with the drop spacing value in mm to execute the printing tasks.

Studies were also carried out to print structure with two drops spacing 100  $\mu\text{m}$  and 50  $\mu\text{m}$ . and drop was dispensed at 150 V and 30  $\mu\text{s}$  pulse width. It was observed that changing the dispensing parameters was affecting the size of the ink drop and the thickness of the printed structure. It was observed that reducing the drop spacing to halve increases the thickness and width of the printed line to about 21 % and 63 % respectively, keeping all the dispensing parameters such as voltage and pulse width same.

The contact resistance was more for a 100  $\mu\text{m}$  drop spacing when oven sintered as may be the probe was penetrating the thickness of the structure and touching the glass surface causing increase in resistance value.

## 7.2 Future Work

Based on the conclusion drawn from this thesis, some future works are described as follows:

Surface modification of the substrate prior to printing was carried out using DI water, acetone and isopropanol. But sometimes poor wetting makes printing of multiple layers difficult. Poor wetting results in high contact angle. Both wetting and adhesion are important for inkjet printed electronics, better surface modification using a plasma cleaning technique may be carried out for future work.

Only a single layer of conductive ink pattern was designed. Future work may include designing of complex structure of multiple layers. Only one nozzle was used to dispense the silver conductive ink for this work. The user interface was designed to control one nozzle and print one layer of structure. User interface may be changed to allow user to switch between the two nozzles and form a conductive and dielectric multiple layer. The ink modification may be done in future as sometimes as poor choice of solvent for ink may lead to some printing challenges such as low throughput and low resolution.

## References

- [1] B. J. De Gans, P. C. Duineveld, and U. S. Schubert, "Inkjet printing of polymers: State of the art and future developments," *Adv. Mater.*, vol. 16, no. 3, pp. 203–213, 2004.
- [2] T. N. Ng et al., "Electrical stability of inkjet-patterned organic complementary inverters measured in ambient conditions," *Appl. Phys. Lett.*, vol. 94, no. 23, pp. 2007–2010, 2009.
- [3] Y. Y. Noh, N. Zhao, M. Caironi, and H. Sirringhaus, "Downscaling of self-aligned, all-printed polymer thin-film transistors," *Nat. Nanotechnol.*, vol. 2, no. 12, pp. 784–789, 2007.
- [4] U. Stachewicz, C. U. Yurteri, J. C. M. Marijnissen, and J. F. Dijkman, "Stability regime of pulse frequency for single event electrospraying," *Appl. Phys. Lett.*, vol. 95, no. 22, pp. 2007–2010, 2009.
- [5] A. Rida, R. Vyas, L. Yang, C. Kruesi, and M. M. Tentzeris, "Low cost inkjet-printing paper-based modules for RFID sensing and wireless applications," *Proc. 38th Eur. Microw. Conf. EuMC 2008*, no. October, pp. 1715–1718, 2008.
- [6] V. Kantola, J. Kulovesi, L. Lahti, R. Lin, M. Zavodchikova, and E. Coatanéa, *Printed Electronics, Now and Future, Bit Bang - Rays to the Future*, ed. Y. In Neuvo & S. Ylönien, 2009, Helsinki University of Technology, Helsinki, Finland, 63-102, 2009.
- [7] C. Sekine, Y. Tsubata, T. Yamada, M. Kitano, and S. Doi, "Recent progress of high performance polymer OLED and OPV materials for organic printed electronics," *Sci. Technol. Adv. Mater.*, vol. 15, no. 3, 2014.
- [8] J. Perelaer, B.-J. de Gans, and U. S. Schubert, "Ink-jet Printing and Microwave Sintering of Conductive Silver Tracks," *Adv. Mater.*, vol. 18, no. 16, pp. 2101–2104, Aug. 2006.
- [9] P. Calvert, "Inkjet printing for materials and devices," *Chem. Mater.*, vol. 13, no. 10, pp. 3299–3305, 2001.
- [10] E. Tekin, P. J. Smith, and U. S. Schubert, "Inkjet printing as a deposition and patterning tool for polymers and inorganic particles," *Soft Matter*, vol. 4, no. 4, pp. 703–713, 2008.
- [11] H. Wijshoff, "The dynamics of the piezo inkjet printhead operation," *Phys. Rep.*, vol. 491, no. 4–5, pp. 77–177, 2010.
- [12] D. Chan, "Development of an Inkjet Printing System on a Flatbed Router," *Master of Applied Science*, University of Waterloo, 2010.
- [13] H. Wijshoff, "The dynamics of the piezo inkjet printhead operation," *Phys. Rep.*, vol. 491, no. 4–5, pp. 77–177, 2010.
- [14] J. Perelaer, P. Kröber, J. T. Delaney, and U. S. Schubert, "Fabrication of two and three-dimensional structures by using inkjet printing," *NIP Digit. Fabr. Conf. 2009 Int. Conf. Digit. Print. Technol.*, pp. 791–794, 2009.
- [15] J. Eggers, "A brief history of drop formation," *University of Bristol*, pp. 1–9, Mar. 2004.
- [16] H. P. Le, "Progress and trends in ink-jet printing technology," *J. Imaging Sci. Technol.*, vol. 42, pp. 49–62, 1998.

- [17] X. Liu, T. J. Tarn, F. Huang, and J. Fan, "Recent advances in inkjet printing synthesis of functional metal oxides," *Particuology*, vol. 19, pp. 1–13, 2015.
- [18] Lord Rayleigh, "On the instability of jets," *Proc. London Math. Soc.*, vol. s1-10, no. 1, pp. 4–13, 1878.
- [19] J. Li, F. Rossignol, and J. Macdonald, "Inkjet printing for biosensor fabrication: Combining chemistry and technology for advanced manufacturing," *Lab Chip*, vol. 15, no. 12, pp. 2538–2558, 2015.
- [20] S. Jung, "Fluid characterisation and drop impact in inkjet printing for organic semiconductor devices," *Doctor of Philosophy, University of Cambridge*, no. April, p. 168, 2011.
- [21] K. K. B. Hon, L. Li, and I. M. Hutchings, "Direct writing technology-Advances and developments," *CIRP Ann. - Manuf. Technol.*, vol. 57, no. 2, pp. 601–620, 2008.
- [22] W. L. Buehner, J. D. Hill, T. H. Williams, and J. W. Woods, "Application of Ink Jet Technology to a Word Processing Output Printer," *IBM J. Res. Dev.*, vol. 21, no. 1, pp. 2–9, 1977.
- [23] J. M. Schneider and S. D. Printing, "Continuous Ink Jet technology explained," *Domino Print. Sci.*, 2012.
- [24] J. Mei, M. R. Lovell, and M. H. Mickle, "Formulation and processing of novel conductive solution inks in continuous inkjet printing of 3-D electric circuits," *IEEE Trans. Electron. Packag. Manuf.*, vol. 28, no. 3, pp. 265–273, 2005.
- [25] A. Loi, "PhD Dissertation title Inkjet printing : technique and applications for organic electronic devices," *Ph.D. Thesis, University of Cagliari*, 2013.
- [26] Y. Xiao, "A Cell Patterning System Based on Thermal Ink-jet Technology," *M.Sc. Thesis, Lehigh University*, 2015.
- [27] T. Xu, J. Jin, C. Gregory, J. J. Hickman, and T. Boland, "Inkjet printing of viable mammalian cells," *Biomaterials*, vol. 26, no. 1, pp. 93–99, 2005.
- [28] E. A. Roth, T. Xu, M. Das, C. Gregory, J. J. Hickman, and T. Boland, "Inkjet printing for high-throughput cell patterning," *Biomaterials*, vol. 25, no. 17, pp. 3707–3715, 2004.
- [29] C. V. Pious and S. Thomas, "Printing on Polymers: Fundamentals and Applications," *Print. Polym. Fundam. Appl.*, no. September 2015, pp. 21–39, 2015.
- [30] H. Reproduction, *Handbook of Digital Imaging*. Chichester, UK: John Wiley & Sons, Ltd, 2015.
- [31] Microdrop Technologies GmbH, "Microdispensing System MD-E-3000," vol. 49, no. October. pp. 1–85, 2016.
- [32] OPTEK Technology Inc., "OPB0AWZ Reflective Object Sensor", OPB0AWZ datasheet, 2016.
- [33] L. Allegro MicroSystems, "A4988 Microstepping Driver", A4988 datasheet, pp. 1–22, 2014 [Revised Jan. 2012].
- [34] S. B. Moore, W. B. Glisson, and M. Yampolskiy, "Implications of Malicious 3D Printer Firmware," *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, pp. 6089–6098, 2017.
- [35] T. H. J. Van Osch, J. Perelaer, A. W. M. De Laat, and U. S. Schubert, "Inkjet printing of narrow

- conductive tracks on untreated polymeric substrates," *Adv. Mater.*, vol. 20, no. 2, pp. 343–345, 2008.
- [36] J. Perelaer, "Microstructures prepared via inkjet printing and embossing techniques," Ph.D. Thesis, Eindhoven University of technology, 2009.
  - [37] H. H. Lee, K. Sen Chou, and K. C. Huang, "Inkjet printing of nanosized silver colloids," *Nanotechnology*, vol. 16, no. 10, pp. 2436–2441, 2005.
  - [38] D. Tobjörk et al., "IR-sintering of ink-jet printed metal-nanoparticles on paper," *Thin Solid Films*, vol. 520, no. 7, pp. 2949–2955, 2012.
  - [39] U. Caglar, "Studies of Inkjet Printing Technology with Focus on Electronic Materials", Tampere University of Technology, vol. 863, no. 2010, 2009.
  - [40] UT Dots Inc., "UTDAgIJ Conductive Silver Nano inks," UTDAgIJ datasheet, March 2018.
  - [41] Y.-Z. Lee, C.-P. Liu, S.-M. Wang, C.-H. Chen, C.-W. Wang, M.-H. Yang, K. Cheng, and J. Chang, "Ink-Jet Printed Passive Electronic Components: Metal-Insulator-Metal Device," in *Pacific Rim Conference on Lasers and Electro-Optics*, 2005. CLEO/Pacific Rim 2005, 2005, pp. 918– 919.
  - [42] A. Sridhar, "An inkjet printing-based process chain for conductive structures on printed circuit board materials," Ph.D. Thesis, University of Twente, Enschede, The Netherlands, 2010.
  - [43] I. A. N. Ink, E. Halonen, T. Viiru, K. Östman, A. L. Cabezas, and M. Mäntysalo, "Oven Sintering Process Optimization for Inkjet-Printed Ag Nanoparticle Ink," *IEEE Trans. COMPONENTS, Packag. Manuf. Technol.*, vol. 3, no. February 2013, pp. 350–356, 2012.

## Appendix A

### Code to design a line.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package shape;
import java.text.DecimalFormat;
import servlet.NewServlet;
/**
 *
 * @author gaura
 */
public class Line {
    public static final String PRINT_CONDUCTIVE_ON = "StartHead(1,1)";
    public static final String PRINT_CONDUCTIVE_OFF = "StartHead(1,0)";

    public String createLine(float space, float line_X_start, float line_Y_start, float line_X_end,
float line_Y_end)
    {
        System.out.println("calling_xAxislinefunctionCall");
        DecimalFormat format = new DecimalFormat("##.000");
        String line_segment = "G1" + " " + "X" + line_X_start + " " + "Y" + line_Y_start + "\r\n";
        float initial_x = line_X_start;
        float initial_y = line_Y_start;
        float angle = (float) Math.atan2(line_Y_end - line_Y_start, line_X_end - line_X_start);
        for (; initial_x < line_X_end || initial_y < line_Y_end;) {
            String x4 = new String();
            initial_x = initial_x + space * (float) Math.cos(angle);
            initial_x = Float.valueOf(format.format(initial_x));
            initial_y = initial_y + space * (float) Math.sin(angle);
            initial_y = Float.valueOf(format.format(initial_y));
            x4 = "G1" + " " + "X" + initial_x + "Y" + initial_y + "\r\n";
            line_segment = line_segment.concat(x4);
        }
        System.out.println(line_segment);
        return line_segment;
    }
}
```



## Appendix B

### Code to design a filled rectangle.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package shape;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author gaura
 */
public class Rectangle {
    static String MOVE = "G1" + " " + "X";
    public static final String PRINTGENERATEON = "StartHead(1,1)";
    public static final String PRINTGENERATEOFF = "StartHead(1,0)";

    public String createAndFillRectangle(float start_angle, float x_start, float x_end, float spacing,
float y_start, float y_end)
    {
        System.out.println("calling_rectfunctionCall");
        String startPoint = new String();
        startPoint = MOVE + x_start + " " + "Y" + y_start + "\r\n";
        float x_startObatined = x_start;
        float y_endObatined = y_end;
        float x_endObatined = x_end;
        DecimalFormat format = new DecimalFormat("##.00");
        float x_prevoius_value = 0;
        for (; y_start < y_endObatined;) {
            for (; x_start < x_end;) {
                String x2 = new String();
                y_end = y_start;
                float x_angle = (float) Math.atan2(y_end - y_start, x_endObatined - x_start);
                float x_rotated_angle = x_angle // + start_angle;
                x_prevoius_value = x_start;
                float x_start_val = x_start + spacing * (float) Math.cos(x_rotated_angle);
                x_start = Float.valueOf(format.format(x_start_val));
                if (x_start > x_end) {
                    break;
                }
                x2 = MOVE + x_start + " " + "Y" + y_start + "\r\n";
            }
        }
    }
}
```

```

        startPoint = startPoint.concat(x2);
        System.out.println("Positive X" + x2);
    }
    float y_angle = 0;
    if (x_start > x_end) {
        x_start = x_prevoius_value;
        y_angle = (float) Math.atan2(y_endObatined - y_start, x_prevoius_value - x_start);
    } else {
        x_start = x_end;
        y_angle = (float) Math.atan2(y_endObatined - y_start, x_end - x_start); }
    float y_rotated_angle = y_angle; // + start_angle;
    // float y_started = y_start;
    float y_start_val = y_start + spacing * (float) Math.sin(y_rotated_angle);
    y_start = Float.valueOf(format.format(y_start_val)); // (y_start_val * 100 / 100);
    for (; x_start > x_startObatined;) {
        String x3 = new String();
        y_end = y_start;
        float angle = (float) Math.atan2(y_end - y_start, x_endObatined - x_start);
        float angle_rotated = angle; // + start_angle;
        x3 = MOVE + x_start + " " + "Y" + y_start + "\r\n";
        float new_x_start = x_start - spacing * (float) Math.cos(angle_rotated);
        x_start = Float.valueOf(format.format(new_x_start));
        startPoint = startPoint.concat(x3);
        System.out.println("Negative X" + x3);
    }
    String x4 = new String();
    x4 = MOVE + x_start + " " + "Y" + y_start + "\r\n";
    startPoint = startPoint.concat(x4);
    System.out.println("X value" + x4);
    float angle = (float) Math.atan2(y_end - y_start, x_end - x_start);
    float y_angles = angle; // + start_angle;
    float val_y_start = y_start + spacing * (float) Math.cos(y_angles);
    y_start = Float.valueOf(format.format(val_y_start)); // (val_y_start * 100 / 100);
    if (y_start > y_endObatined) {
        break;
    }
    String x5 = new String();
    x5 = MOVE + x_start + " " + "Y" + y_start + "\r\n";
    startPoint = startPoint.concat(x5);
    System.out.println("X New_value" + x5);
}
System.out.println("Final movement" + startPoint);
System.out.println("END_rectfunctionCall");
return startPoint;
}
}

```

## Appendix C

### Code to design a filled circle with vertical lines.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package shape;
import java.text.DecimalFormat;

/**
 *
 * @author gaura
 */
public class Circle {

    public static final String PRINTGENERATEON = "StartHead(1,1)";
    public static final String PRINTGENERATEOFF = "StartHead(1,0)";

    public String createCircle(float space, float circle_X_centre, float circle_Y_centre, float radius, float
start_angle, String isClock)
    {
        System.out.println("calling_circlefunctionCall");
        float angle = 0f;
        String x_circle = new String();
        float circle_X_coord = 0;
        float circle_Y_coord = 0;
        boolean isDropCircle = false;
        DecimalFormat format = new DecimalFormat("##.00");
        for (float i = 0; i < 36; i++) {
            String x1 = new String();
            float ten_degree_point_spacing = (float) (2 * Math.PI * i / 36);
            start_angle = (float) Math.toRadians(start_angle);
            start_angle = ten_degree_point_spacing + start_angle;
            if ("Clockwise".equalsIgnoreCase(isClock)) {
                angle = -start_angle;
            } else {
                angle = start_angle;
            }
            circle_X_coord = (circle_X_centre + radius * (float) Math.cos(angle));
            circle_Y_coord = (circle_Y_centre + radius * (float) Math.sin(angle));
            x1 = "G1" + "X" + (Float.valueOf(format.format(circle_X_coord))) + "Y" +
(Float.valueOf(format.format(circle_Y_coord)));//+ ":" + PRINTGENERATEON + ":" +
PRINTGENERATEOFF + "\r\n";
            String x3 = isDropCircle ? x1 + ":" + PRINTGENERATEON + ":" + PRINTGENERATEOFF +
"\r\n" : x1 + "\r\n";
            System.out.println("Circle Coordinates" + x3);
            x_circle = x_circle.concat(x3);
        }
    }
}
```

```

    // isDropCircle = true;
}
// first calculate two X extremes points
//
float X_one_extreme_point = circle_X_centre + radius * (float) Math.cos(0);

float X_another_extreme_point = circle_X_centre - radius * (float) Math.cos(0);
float y_coordinate_at_zero = (circle_Y_centre + radius * (float) Math.sin(0));
circle_Y_coord = y_coordinate_at_zero;
float reduce_x_coordinate = X_one_extreme_point - space;// - circle_X_centre;
float value_x = 0;
float value_y = 0;
if (reduce_x_coordinate > circle_X_centre) {
    value_x = reduce_x_coordinate - circle_X_centre;
} else {
    value_x = circle_X_centre - reduce_x_coordinate;
}
if (circle_Y_coord > circle_Y_centre) {
    value_y = circle_Y_coord - circle_Y_centre;
} else {
    value_y = circle_Y_centre - circle_Y_coord;
}
float calculate_distance = (float) Math.sqrt(Math.pow(value_x, 2) + Math.pow(value_y, 2));
float theta = (float) Math.acos(((double) ((calculate_distance) / radius)));
float tot_angle = theta;// + start_angle_original;
System.out.println("Angle theta" + tot_angle);
float pos_Y_coordinate = radius * (float) Math.sin(tot_angle) + circle_Y_centre;
System.out.println("pos_Y_coordinate based on thetha" + pos_Y_coordinate);
float Y_one_extreme_point = pos_Y_coordinate;
float Y_another_extreme_point = circle_Y_centre - radius * (float) Math.sin(tot_angle);
boolean isYGreater = false;
boolean isDropNeed = false;
for (; Math.round(X_one_extreme_point) > Math.round(X_another_extreme_point);) {
    //X axis movement till it reaches the other extreme point
    if (Y_one_extreme_point > circle_Y_centre) {
        // this is for positive Y
        isYGreater = true;
        for (; Y_one_extreme_point > (Y_another_extreme_point);) {
            // Y axis movement
            String x2 = new String();
            // Y_first_point = Y_one_extreme_point;
            x2 = "G1" + "X" + (Float.valueOf(format.format(reduce_x_coordinate))) + "Y" +
            (Float.valueOf(format.format(Y_one_extreme_point)));// + "\r\n";
            String x3 = isDropNeed ? x2 + ":@" + PRINTGENERATEON + ":@" + PRINTGENERATEOFF
+ "\n" : x2 + "\n";
            System.out.println("positive Y" + x3);
            x_circle = x_circle.concat(x3);
            float reduced_pos_Y_coordinate = Y_one_extreme_point - space;
            Y_one_extreme_point = reduced_pos_Y_coordinate;
            // isDropNeed = true;
        }
    }
}

```

```

String x6 = new String();
x6 = "G1" + "X" + (Float.valueOf(format.format(reduce_x_coordinate))) + "Y" +
(Float.valueOf(format.format(Y_another_extreme_point)));// + "\r\n";
String x7 = isDropNeed ? x6 + ":" + PRINTGENERATEON + ":" + PRINTGENERATEOFF +
"\n" : x6 + "\n";
System.out.println("positive Y" + x7);
x_circle = x_circle.concat(x7);
} else {
//this is for negative Y
// starts from the point where it ends
isYGreater = false;
isDropNeed = false;
for (; (Y_one_extreme_point) < (Y_another_extreme_point);) {
// Y axis movement
String x3 = new String();
// Y_first_point = Y_one_extreme_point;
x3 = "G1" + "X" + (Float.valueOf(format.format(reduce_x_coordinate))) + "Y" +
(Float.valueOf(format.format(Y_one_extreme_point)));// + "\r\n";
String x8 = isDropNeed ? x3 + ":" + PRINTGENERATEON + ":" + PRINTGENERATEOFF
+ "\n" : x3 + "\n";
System.out.println("negative Y" + x8);
x_circle = x_circle.concat(x8);
float reduced_pos_Y_coordinate = Y_one_extreme_point + space;
Y_one_extreme_point = reduced_pos_Y_coordinate;
// isDropNeed = true;
}
}
float reduced_x_coordinate = reduce_x_coordinate - space;
reduce_x_coordinate = reduced_x_coordinate;
float x_value = 0;
float y_value = 0;
if (reduce_x_coordinate > circle_X_centre) {
x_value = reduce_x_coordinate - circle_X_centre;
} else {
x_value = circle_X_centre - reduce_x_coordinate;
}

if (circle_Y_coord > circle_Y_centre) {
y_value = circle_Y_coord - circle_Y_centre;
} else {
y_value = circle_Y_centre - circle_Y_coord;
}

calcuete_distance = (float) Math.sqrt(Math.pow(x_value, 2) + Math.pow(y_value, 2));
theta = (float) Math.acos((double) ((calcuete_distance) / radius));
float Y_one_point = 0;
float Y_another_point = 0;
float total_angle = theta;// + start_angle_original;
if (isYGreater) {
//starts from Xc minus point
Y_one_extreme_point = radius * (float) Math.sin(total_angle);

```

```

        Y_one_point = circle_Y_centre - Y_one_extreme_point;
        Y_another_point = circle_Y_centre + Y_one_extreme_point;
    } else {
        //starts from Xc plus point
        Y_one_extreme_point = radius * -(float) Math.sin(total_angle);
        Y_one_point = circle_Y_centre - Y_one_extreme_point;
        Y_another_point = circle_Y_centre + Y_one_extreme_point;

    }
    Y_one_extreme_point = Y_one_point;
    Y_another_extreme_point = Y_another_point;
    X_one_extreme_point = reduce_x_coordinate;
}
return x_circle;
}
}

```

## Appendix D

### Setting up the connection in servlet Context while deploying the server.

The motor controller thread is asked to wait until “ok” message is received from motor controller and dispenser controller thread responds properly. The dispenser controller thread is asked to wait until motor controller responds properly. The main thread waits to complete the entire tasks from these threads (till the last line of the instruction file is read).

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package contextListener;

import java.util.HashMap;
import java.util.Map;
import javax.servlet.ServletContext;
import jssc.SerialPort;
import jssc.SerialPortEvent;
import jssc.SerialPortEventListener;
import jssc.SerialPortException;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

/**
 *
 * @author gaura
 */
public class MyServletContextListener implements ServletContextListener {
    ServletContext context;
    public SerialPort serialMotorPort;
    public SerialPort serialNozzlePort;
    public String str[] = {"voltage", "pulseWidth", "isTemp", "freq", "drop"};
    private Map<String, String> map = new HashMap<>();
    public static boolean isCheck = false;
    public Thread motorThread = null;
    public Thread nozzleThread = null;
    public static final Object LOCKED = new Object();
    public static final Object NOZZLELOCKED = new Object();
    public static final Object NOZZLELOCKEDECHOBUSY = new Object();
```

```

public static final Object LOCKEDLASTLINE = new Object();
static boolean ready = false;
static boolean readyNozzle = false;
static boolean readyNozzleEchoBusy = false;
static boolean isFileBlank = false;
static String readData = new String();
static String readDataNozzle = new String();
static String readDataMotorEchoBusy = new String();
static String readLastLine = new String();
static String receivedData;
// private SerialPortMotorReader myThread = null;
// public static String receivedData = null;

@Override
public void contextDestroyed(ServletContextEvent arg0) {
    try {
        //Notification that the servlet cwqontext is about to be shut down.
        context = arg0.getServletContext();
        serialMotorPort = (SerialPort) context.getAttribute("portObject");
        serialNozzlePort = (SerialPort) context.getAttribute("portObject1");
        if (context != null && serialMotorPort != null && serialNozzlePort != null &&
serialMotorPort.isOpened() && serialNozzlePort.isOpened()) {
            serialMotorPort.removeEventListener();
            serialMotorPort.closePort();
            serialNozzlePort.removeEventListener();
            serialNozzlePort.closePort();
        }
        // motorThread.destroy();

//         if (scheduler != null) {
//             scheduler.shutdown();
//             // context.getScheduler().shutdown();
//         }
    } catch (SerialPortException ex) {
        System.out.println(ex);
    }
}

@Override
public void contextInitialized(ServletContextEvent arg0) {
    // do all the tasks that you need to perform just after the server starts
    System.out.println("Context Created");
    context = arg0.getServletContext();

    serialMotorPort = new SerialPort("COM3");
    serialNozzlePort = new SerialPort("COM7");
    try {
        serialMotorPort.openPort();

```



```

        serialNozzlePort.openPort();
        serialMotorPort.setParams(250000, 8, 1, 0);
        serialNozzlePort.setParams(19200, 8, 1, 0);
        int mask = SerialPort.MASK_RXCHAR + SerialPort.MASK_CTS + SerialPort.MASK_DSR;
        //Set the prepared mask
        serialMotorPort.setEventsMask(mask);
        serialNozzlePort.setEventsMask(mask);
        // context = arg0.getServletContext();
        context.setAttribute("portObject", serialMotorPort);
        context.setAttribute("portObject1", serialNozzlePort);
        // context.setAttribute("semaphore", semaphore);
        serialMotorPort.addEventListener((SerialPortEventListener) new
MyServletContextListener.SerialPortMotorReader());
        serialNozzlePort.addEventListener((SerialPortEventListener) new
MyServletContextListener.SerialPortNozzleReader());
    } catch (SerialPortException ex) {
        context.setAttribute("SerialPortException", ex);
        System.out.println(ex);
    }
}

public class SerialPortMotorReader implements SerialPortEventListener {

    // @Override
    public StringBuffer sb = new StringBuffer();
    @Override
    public void serialEvent(SerialPortEvent event) {
        if (event.isRXCHAR() && event.getEventValue() > 0) {
            try {
                byte buffer[] = serialMotorPort.readBytes();
                if (!"\n".equalsIgnoreCase(new String(buffer))) {
                    //sb = new StringBuffer();
                    sb.append(new String(buffer));
                    System.out.println("Response from Motor inside serial Event" + new String(buffer));
                    context.setAttribute("message", sb);
                    // isCalledOnce = true;
                }
                if ("echo:busy: processing\n".equalsIgnoreCase(new String(buffer))) {
                    // setDataMotorEchoBusyOrOk(new String(buffer));
                }
                // if ("ok\n".equalsIgnoreCase(new String(buffer))) {
                setDataMotor(new String(buffer));
                //setDataNozzle(">");
                // }
                if ("ok\nEMERGENCY STOP\n".equalsIgnoreCase(new String(buffer))
                    /* || "echo:Unknown command:\n".equalsIgnoreCase(new String(buffer)) */) {
                    // this is where everything should stop
                    serialNozzlePort.writeString("StartHead(1,0)" + "\r"); //stop the head
                    serialNozzlePort.removeEventListener();
                }
            }
        }
    }
}

```

```

        serialNozzlePort.closePort();
        serialMotorPort.writeString("M81" + "\r\n");
        byte buffer1[] = serialMotorPort.readBytes(3);
        if (!"ok\n".equalsIgnoreCase(new String(buffer1))) {
            serialMotorPort.removeEventListener();
            serialMotorPort.closePort();
            System.out.println("EVERYTHING STOPPED");
        }
    }
}

} catch (SerialPortException ex) {
    context.setAttribute("SerialPortException", ex);
    System.out.println(ex);
}
}
else if (event.isCTS()) {
    if (event.getEventValue() == 1) {
        System.out.println("CTS - ON");
    } else {
        System.out.println("CTS - OFF");
    }
} else if (event.isDSR()) {
    if (event.getEventValue() == 1) {
        System.out.println("DSR - ON");
    } else {
        System.out.println("DSR - OFF");
    }
}
}
}

public class SerialPortNozzleReader implements SerialPortEventListener {

    // public String receivedData1 = new String();
    public StringBuffer sb1 = new StringBuffer();
    boolean isCalledOnceNozzleParam = false;

    @Override
    public void serialEvent(SerialPortEvent event) {
        if (event.isRXCHAR() && event.getEventValue() > 0) {
            try {
                receivedData = serialNozzlePort.readString();
                System.out.println("Response from Nozzle inside serial Event" + receivedData);
                sb1.append(receivedData);
                setDataNozzle(receivedData);
            } catch (SerialPortException ex) {
                context.setAttribute("SerialPortException", ex);
                System.out.println(ex);
            }
        }
    }
}

```

```

    } else if (event.isCTS()) {
        if (event.getEventValue() == 1) {
            System.out.println("CTS - ON");
        } else {
            System.out.println("CTS - OFF");
        }
    } else if (event.isDSR()) {
        if (event.getEventValue() == 1) {
            System.out.println("DSR - ON");
        } else {
            System.out.println("DSR - OFF");
        }
    }
}
}

public static String getDataMotor(boolean isReady) {
    synchronized (LOCKED) {
        ready = isReady;
        while (ready == false) {
            try {
                System.out.println("inside getDataMotor" + readData);
                LOCKED.wait();
            } catch (InterruptedException ex) {
                System.out.println("InterruptedException" + ex);
            }
        }
        String temp = readData;
        readData = "";
        return temp;
    }
}

public static void setDataMotor(String data) {
    synchronized (LOCKED) {
        readData = data;
        System.out.println("inside setDataMotor" + readData);
        if ("ok\n".equalsIgnoreCase(readData) || "echo:busy: processing\n".equalsIgnoreCase(readData))
    {
        System.out.println("OK to proceed" + readData);
        ready = true;
        LOCKED.notifyAll();
        // setDataNozzle(receivedData);
    }
}
}

public static String getDataNozzle(boolean isReadyNozzle) {

```

```

synchronized (NOZZLELOCKED) {
    readyNozzle = isReadyNozzle;
    while (readyNozzle == false) {
        try {
            System.out.println("inside getDataNozzle" + readDataNozzle);
            //System.out.println("motor result" + readData);
            NOZZLELOCKED.wait();
        } catch (InterruptedException ex) {
            System.out.println("InterruptedException" + ex);
        }
    }
    String temp = readDataNozzle;
    readDataNozzle = "";
    return temp;
}

public static void setDataNozzle(String data) {
    synchronized (NOZZLELOCKED) {
        readDataNozzle = data;
        System.out.println("inside setDataNozzle" + readDataNozzle);
        // System.out.println("Motor Response" + readData);
        if (readDataNozzle != null && ">".contentEquals(readDataNozzle) /*&&
("ok\n".equalsIgnoreCase(readData))*/) {
            System.out.println("inside setData Nozzle" + readDataNozzle);
            readyNozzle = true;
            NOZZLELOCKED.notifyAll();
        }
    }
}

public static String getFileLastLine() {
    synchronized (LOCKEDLASTLINE) {
        while (isFileBlank == false) {
            try {
                System.out.println("inside getFileLastLine" + readLastLine);
                LOCKEDLASTLINE.wait();
            } catch (InterruptedException ex) {
                System.out.println("InterruptedException" + ex);
            }
        }
        String temp = readLastLine;
        readLastLine = "";
        return temp;
    }
}

public static void setFileLastLine(String data) {

```

```
synchronized (LOCKEDLASTLINE) {  
    readLastLine = data;  
    System.out.println("inside setFileLastLine" + readLastLine);  
    if ("".equalsIgnoreCase(readLastLine)) {  
        System.out.println("OK to proceed in setFileLastLine" + readLastLine);  
        isFileBlank = true;  
        LOCKEDLASTLINE.notifyAll();  
    }  
}  
}  
}
```

## Appendix E

This servlet file receives the request from user interface and transfers it to the service class, receives the response from the service class and pass it to the user interface.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package servlet;

import contextListener.MyServletContextListener;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.LineNumberReader;
import java.io.PrintWriter;
import java.net.URL;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import jssc.SerialPort;
import jssc.SerialPortException;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
import shape.Circle;
import shape.Dot;
import shape.Line;
import shape.Rectangle;

/**
 *
 * @author gaura
 */
public class NewServlet extends HttpServlet {
```

```

// SerialPort serialMotorPort;
// SerialPort serialNozzlePort;
StringBuffer sb = new StringBuffer();
StringBuffer sb1 = new StringBuffer();
private final String UPLOAD_DIRECTORY = "C:/Users/gaura/Desktop/";
public static final String PRINTGENERATEON = "StartHead(1,1)";
public static final String PRINTGENERATEOFF = "StartHead(1,0)";
static String move = "G1" + " " + "X";
public static final String FILENAME = "C:\\Users\\gaura\\Desktop\\myFile.txt";
public static final String UPLOAD_FILENAME = "C:\\Users\\gaura\\Desktop\\uploadFile.txt";
File file = new File(FILENAME);
File upload_file = new File(UPLOAD_FILENAME);
BufferedWriter outputWriter;
BufferedReader bufferedReader;
BufferedWriter outputWriter1;
BufferedReader bufferedReader1;
BufferedReader br;

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    doPost(request, response);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    ServletContext context = getServletContext();
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    float spacing = request.getParameter("space") != null ?
Float.parseFloat(request.getParameter("space")) : 0;
    String arc_start = (request.getParameter("arc_start") != null &&
!"".equals(request.getParameter("arc_start"))) ? request.getParameter("arc_start") : "";
    String arc_end = (request.getParameter("arc_end") != null &&
!"".equals(request.getParameter("arc_end"))) ? request.getParameter("arc_end") : "";
    String arc_center = (request.getParameter("arc_center") != null &&
!"".equals(request.getParameter("arc_center"))) ? request.getParameter("arc_center") : "";
    String createStructure = new String();
    if (null != request.getParameter("dotSubmit")) {
        // Login form submitted.
        String start_coord = (request.getParameter("start_coord") != null &&
!"".equals(request.getParameter("start_coord"))) ? (request.getParameter("start_coord")) : "";
        String[] arc_start_array = !"".equalsIgnoreCase(start_coord) ? start_coord.split(",") : null;
        if (arc_start_array != null /*&& arc_end_array != null*/) {
            float line_X_start = Float.parseFloat(arc_start_array[0]);
            float line_Y_start = Float.parseFloat(arc_start_array[1]);

```

```

        createStructure = new Dot().createDot(line_X_start, line_Y_start);
    } else {
        System.out.println("Input Not received in correct format");
    }

    try {
        if (!file.exists()) {
            file.createNewFile();
        }
        outputWriter = new BufferedWriter(new FileWriter(file));
        outputWriter.write(createStructure);
        outputWriter.newLine();
        outputWriter.flush();
    } catch (IOException ex) {
        System.out.println("IOException while writing" + ex);
        request.setAttribute("message", "rectfunctionCall_IOException" + ex);
    }

    int lines = lineCount();
    readFile(FILENAME, request, response, context, lines);
} else if (null != request.getParameter("lineSubmit")) {
    // Something else submitted.
    // float space = (request.getParameter("space") != null &&
    !"".equals(request.getParameter("space"))) ? Float.parseFloat(request.getParameter("space")) : 1;
    String start_coord = (request.getParameter("start_coord") != null &&
    !"".equals(request.getParameter("start_coord"))) ? (request.getParameter("start_coord")) : "";
    String end_coord = (request.getParameter("end_coord") != null &&
    !"".equals(request.getParameter("end_coord"))) ? (request.getParameter("end_coord")) : "";

    // String[] circle_start_array = !"".equalsIgnoreCase(start_angle) ? start_angle.split(",") : null;
    String[] arc_start_array = !"".equalsIgnoreCase(start_coord) ? start_coord.split(",") : null;
    String[] arc_end_array = !"".equalsIgnoreCase(end_coord) ? end_coord.split(",") : null;
    if (arc_start_array != null && arc_end_array != null) {
        float line_X_start = Float.parseFloat(arc_start_array[0]);
        float line_Y_start = Float.parseFloat(arc_start_array[1]);
        float line_X_end = Float.parseFloat(arc_end_array[0]);
        float line_Y_end = Float.parseFloat(arc_end_array[1]);
        createStructure = new Line().createLine(spacing, line_X_start, line_Y_start, line_X_end,
line_Y_end);
    } else {
        System.out.println("Input Not received in correct format");
    }

    try {
        if (!file.exists()) {
            file.createNewFile();
        }
        outputWriter = new BufferedWriter(new FileWriter(file));
        outputWriter.write(createStructure);
        outputWriter.newLine();

```



```

        outputWriter.flush();
    } catch (IOException ex) {
        System.out.println("IOException while writing" + ex);
        request.setAttribute("message", "rectfunctionCall_IOException" + ex);
    }
    int lines = lineCount();
    readFile(FILENAME, request, response, context, lines);
} else if (null != request.getParameter("rectSubmit")) {
    // Something else submitted.
    float start_angle = (request.getParameter("start_angle") != null &&
!"".equals(request.getParameter("start_angle"))) ?
Float.parseFloat(request.getParameter("start_angle")) : 0;
    String start_coord = (request.getParameter("start_coord") != null &&
!"".equals(request.getParameter("start_coord"))) ? (request.getParameter("start_coord")) : "";
    String end_coord = (request.getParameter("end_coord") != null &&
!"".equals(request.getParameter("end_coord"))) ? (request.getParameter("end_coord")) : "";
    // String[] circle_start_array = !"".equalsIgnoreCase(start_angle) ? start_angle.split(",") : null;
    String[] arc_start_array = !"".equalsIgnoreCase(start_coord) ? start_coord.split(",") : null;
    String[] arc_end_array = !"".equalsIgnoreCase(end_coord) ? end_coord.split(",") : null;
    if (arc_start_array != null && arc_end_array != null) {
        float line_X_start = Float.parseFloat(arc_start_array[0]);
        float line_Y_start = Float.parseFloat(arc_start_array[1]);
        float line_X_end = Float.parseFloat(arc_end_array[0]);
        float line_Y_end = Float.parseFloat(arc_end_array[1]);
        createStructure = new Rectangle().createAndFillRectangle(start_angle, line_X_start,
line_X_end, spacing, line_Y_start, line_Y_end);
    } else {
        System.out.println("Input Not received in correct format");
    }
}
try {
    if (!file.exists()) {
        file.createNewFile();
    }
    outputWriter = new BufferedWriter(new FileWriter(file));
    outputWriter.write(createStructure);
    outputWriter.newLine();
    outputWriter.flush();
} catch (IOException ex) {
    System.out.println("IOException while writing" + ex);
    request.setAttribute("message", "rectfunctionCall_IOException" + ex);
}
int lines = lineCount();
readFile(FILENAME, request, response, context, lines);
} else if (null != request.getParameter("circleSubmit")) {
    // Something else submitted.
    String isCheck = request.getParameter("direction");
    float radius = (request.getParameter("radius") != null &&
!"".equals(request.getParameter("radius"))) ? Float.parseFloat(request.getParameter("radius")) : 0;

```

```

        float start_angle = (request.getParameter("start_angle") != null &&
!"".equals(request.getParameter("start_angle"))) ?
Float.parseFloat(request.getParameter("start_angle")) : 0;
        String circle_centre = (request.getParameter("circle_centre") != null &&
!"".equals(request.getParameter("circle_centre"))) ? request.getParameter("circle_centre") : "";
        String[] circle_centre_array = !"".equalsIgnoreCase(circle_centre) ? circle_centre.split(",") : null;
        float circle_X_centre = 0f;
        float circle_Y_centre = 0f;
        if (circle_centre_array != null) {
            circle_X_centre = Float.parseFloat(circle_centre_array[0]);
            circle_Y_centre = Float.parseFloat(circle_centre_array[1]);
        } else {
            System.out.println("Input Not received in correct format");
        }
        createStructure = new Circle().createCircle(spacing, circle_X_centre, circle_Y_centre, radius,
start_angle, isCheck);
        try {
            if (!file.exists()) {
                file.createNewFile();
            }
            outputWriter = new BufferedWriter(new FileWriter(file));
            outputWriter.write(createStructure);
            outputWriter.newLine();
            outputWriter.flush();
        } catch (IOException ex) {
            System.out.println("IOException while writing" + ex);
            request.setAttribute("message", "rectfunctionCall_IOException" + ex);
        }
        int lines = lineCount();
        readFile(FILENAME, request, response, context, lines);
    } else if (null != request.getParameter("param1")) {
        System.out.println("Ajax Suuccesss");
        readGCodeInstruct(request, response, "G28 X0", out, context);
    } else if (null != request.getParameter("param2")) {
        System.out.println("Ajax Suuccesss");
        readGCodeInstruct(request, response, "G28 Z0", out, context);
    } else if (null != request.getParameter("param4")) {
        System.out.println("Ajax Suuccesss");
        readGCodeInstruct(request, response, "G28 Y0", out, context);
    } else if (null != request.getParameter("param3")) {
        System.out.println("Ajax Suuccesss");
        readGCodeInstruct(request, response, "G28", out, context);
    } else if (null != request.getParameter("param5")) {
        System.out.println("Ajax Suuccesss");
        readGCodeInstruct(request, response, "G1 Y-100", out, context);
    } else if (null != request.getParameter("param6")) {
        System.out.println("Ajax Suuccesss");
        readGCodeInstruct(request, response, "G1 Y-10", out, context);
    }

```

```

} else if (null != request.getParameter("param7")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 Y-1", out, context);
} else if (null != request.getParameter("param8")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 Y-0.1", out, context);
} else if (null != request.getParameter("param9")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 Y100", out, context);
} else if (null != request.getParameter("param10")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 Y10", out, context);
} else if (null != request.getParameter("param11")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 Y1", out, context);
} else if (null != request.getParameter("param12")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 Y0.1", out, context);
} else if (null != request.getParameter("param13")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 X100", out, context);
} else if (null != request.getParameter("param14")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 X10", out, context);
} else if (null != request.getParameter("param15")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 X1", out, context);
} else if (null != request.getParameter("param16")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 X0.1", out, context);
} else if (null != request.getParameter("param17")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 X-100", out, context);
} else if (null != request.getParameter("param18")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 X-10", out, context);
} else if (null != request.getParameter("param19")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 X-1", out, context);
} else if (null != request.getParameter("param20")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 X-0.1", out, context);
} else if (null != request.getParameter("param21")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1" + " " + "Z10", out, context);
} else if (null != request.getParameter("param22")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1 Z1", out, context);
}

```

```

} else if (null != request.getParameter("param23")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1" + " " + "Z0.1", out, context);
} else if (null != request.getParameter("param24")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1" + " " + "Z-10", out, context);
} else if (null != request.getParameter("param25")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1" + " " + "Z-1", out, context);
} else if (null != request.getParameter("param26")) {
    System.out.println("Ajax Suucesss");
    readGCodeInstruct(request, response, "G1" + " " + "Z-0.1", out, context);
} else if (null != request.getParameter("sendInstruct")) {
    System.out.println("Ajax Suucesss");
    String instruct = request.getParameter("instruct");
    readGCodeInstruct(request, response, instruct, out, context);
} //process only if its multipart content
else if (null != request.getParameter("pulseOn")) {
    System.out.println("Ajax Suucesss");
    String instruct = PRINTGENERATEON;
    pulseGenerationOn(request, response, instruct, out, context);
} else if (null != request.getParameter("pulseOff")) {
    System.out.println("Ajax Suucesss");
    String instruct = PRINTGENERATEOFF;
    pulseGenerationOff(request, response, instruct, out, context);
} else if (ServletFileUpload.isMultipartContent(request)) {
    String name = null;
    try {
        List<FileItem> multipart = new ServletFileUpload(new
DiskFileItemFactory()).parseRequest(request);
        for (FileItem item : multipart) {
            if (!item.isFormField()) {
                name = new File(item.getName()).getName();
                item.write(new File(UPLOAD_DIRECTORY + File.separator + name));
                InputStream stream = item.getInputStream();
                URL url = getClass().getClassLoader().getResource(name);
            }
        }
        //File uploaded successfully
        System.out.println("File Uploaded Successfully");
        String local_directory = UPLOAD_DIRECTORY + File.separator + name;
        int lines = lineCount();
        readStructureFromFile(local_directory, request, response, context, lines);
        // request.getRequestDispatcher("/index.jsp").forward(request, response);
    } catch (Exception ex) {
        out.println(ex);
        request.setAttribute("message", "Error in uploading file" + ex);
    }
}

```

```

    }
    try {
        request.getRequestDispatcher("/index.jsp").forward(request, response);
    } catch (IOException | ServletException ex) {
        request.setAttribute("message", "yAxislinefunctionCall_ServletException" + ex);
    }
}

```

```

protected void readGCodeInstruct(HttpServletRequest request, HttpServletResponse response, String
line, PrintWriter out, ServletContext context) {

```

```

    try {
        SerialPort serialMotorPort = (SerialPort) context.getAttribute("portObject");
        if (serialMotorPort != null) {
            serialMotorPort.writeString(line + "\r\n");
        }
        sb = (StringBuffer) context.getAttribute("message");
        request.setAttribute("message", sb);
        System.out.println("message" + sb);
    } catch (SerialPortException ex) {
        System.out.println(ex);
        request.setAttribute("message", "SerialPortException" + ex);
    }
}

```

```

public void pulseGenerationOn(HttpServletRequest request, HttpServletResponse response, String
start, PrintWriter out, ServletContext context) {

```

```

    try {
        SerialPort serialNozzlePort = (SerialPort) context.getAttribute("portObject1");
        if (serialNozzlePort != null) {
            serialNozzlePort.writeString(start + "\r");
        }
        sb1 = (StringBuffer) context.getAttribute("message");
        request.setAttribute("message", sb1);
        System.out.println("message" + sb1);
    } catch (SerialPortException ex) {
        System.out.println(ex);
        // request.setAttribute("message1", "SerialPortException" + ex);
    }
}

```

```

protected void pulseGenerationOff(HttpServletRequest request, HttpServletResponse response, String
end, PrintWriter out, ServletContext context) {

```

```

    try {
        SerialPort serialNozzlePort = (SerialPort) context.getAttribute("portObject1");
        if (serialNozzlePort != null) {
            serialNozzlePort.writeString(end + "\r");
        }
        sb1 = (StringBuffer) context.getAttribute("message");

```

```

        request.setAttribute("message", sb1);
        System.out.println("message" + sb1);
    } catch (SerialPortException ex) {
        System.out.println(ex);
        request.setAttribute("message1", "SerialPortException" + ex);
    }
}

private int lineCount() {
    LineNumberReader lineNumberReader;
    int lines = 0;
    try {
        lineNumberReader = new LineNumberReader(new FileReader(file));
        lineNumberReader.skip(Long.MAX_VALUE);
        lines = lineNumberReader.getLineNumber();
        lineNumberReader.close();
    } catch (FileNotFoundException ex) {
        System.out.print("FileNotFoundException" + ex);
    } catch (IOException ex) {
        System.out.print("IOException in reading line number" + ex);
    }
    return lines;
}

public void readStructureFromFile(String fileName, HttpServletRequest request, HttpServletResponse
response, ServletContext context, int lineNumber) {
    String line = null;
    FileReader fileReader = null;
    String structure = new String();
    try {
        fileReader = new FileReader(upload_file);
    } catch (FileNotFoundException ex) {
        System.out.println("FileNotFoundException" + ex);
    }
    bufferedReader1 = new BufferedReader(fileReader);
    try {
        while ((line = bufferedReader1.readLine()) != null) {
            String namepass[] = line.split(":");
            String move_Segmnent = namepass[0];
            // try {
            System.out.println("Move segment by segment" + " " + move_Segmnent);
            if (!"".equalsIgnoreCase(move_Segmnent)) {
                if ("L".equalsIgnoreCase(move_Segmnent)) {
                    //this is for line
                    //L:2:10,10:30,30
                    // if this is there then first this is drawn and completed and then proceed with some other
structure
                    String drop_spacing = namepass[1];

```

```

        String line_start_coord = namepass[2];
        String line_end_coord = namepass[3];
        float spacing = drop_spacing != null ? Float.parseFloat(drop_spacing) : 0;
        String[] line_start_coord_array = !"".equalsIgnoreCase(line_start_coord) ?
line_start_coord.split(",") : null;
        String[] line_end_coord_array = !"".equalsIgnoreCase(line_end_coord) ?
line_end_coord.split(",") : null;
        if (line_start_coord_array != null && line_end_coord_array != null) {
            float line_X_start = Float.parseFloat(line_start_coord_array[0]);
            float line_Y_start = Float.parseFloat(line_start_coord_array[1]);
            float line_X_end = Float.parseFloat(line_end_coord_array[0]);
            float line_Y_end = Float.parseFloat(line_end_coord_array[1]);
            structure = new Line().createLine(spacing, line_X_start, line_Y_start, line_X_end,
line_Y_end);
        } else {
            System.out.println("Input of Line Not received in correct format");
        }
        try {
            if (!file.exists()) {
                file.createNewFile();
            }
            outputWriter1 = new BufferedWriter(new FileWriter(file));
            outputWriter1.write(structure);
            outputWriter1.newLine();
            outputWriter1.flush();
        } catch (IOException ex) {
            System.out.println("IOException while writing" + ex);
            request.setAttribute("message", "rectfunctionCall_IOException" + ex);
        }
        int lines = lineCount();
        readFile(FILENAME, request, response, context, lines);
    } else if ("C".equalsIgnoreCase(move_Segmnent)) {
        //this is for circle
        // C:2,2:20:0:Clockwise:1
        String circle_centre_coord = namepass[1];
        String circle_radius = namepass[2];
        String circle_start_angle = namepass[3];
        String isCheck = namepass[4];
        String drop_spacing = namepass[5];
        float spacing = drop_spacing != null ? Float.parseFloat(drop_spacing) : 0;
        float radius = drop_spacing != null ? Float.parseFloat(circle_radius) : 0;
        float start_angle = drop_spacing != null ? Float.parseFloat(circle_start_angle) : 0;
        String[] circle_centre_array = !"".equalsIgnoreCase(circle_centre_coord) ?
circle_centre_coord.split(",") : null;
        float circle_X_centre = 0f;
        float circle_Y_centre = 0f;
        if (circle_centre_array != null) {
            circle_X_centre = Float.parseFloat(circle_centre_array[0]);

```

```

        circle_Y_centre = Float.parseFloat(circle_centre_array[1]);
    } else {
        System.out.println("Input of Circle Not received in correct format");
    }
    structure = new Circle().createCircle(spacing, circle_X_centre, circle_Y_centre, radius,
start_angle, isCheck);
    try {
        if (!file.exists()) {
            file.createNewFile();
        }
        outputWriter = new BufferedWriter(new FileWriter(file));
        outputWriter.write(structure);
        outputWriter.newLine();
        outputWriter.flush();
    } catch (IOException ex) {
        System.out.println("IOException while writing" + ex);
        request.setAttribute("message", "rectfunctionCall_IOException" + ex);
    }
    int lines = lineCount();
    readFile(FILENAME, request, response, context, lines);

} else if ("R".equalsIgnoreCase(move_Segment)) {
    // this is for rectangle
    //R:0,0:40,40:2
    String rect_start_coord = namepass[1];
    String rect_end_coord = namepass[2];
    String drop_spacing = namepass[3];
    float spacing = drop_spacing != null ? Float.parseFloat(drop_spacing) : 0;
    String[] rect_start_coord_array = !"".equalsIgnoreCase(rect_start_coord) ?
rect_start_coord.split(",") : null;
    String[] rect_end_coord_array = !"".equalsIgnoreCase(rect_end_coord) ?
rect_end_coord.split(",") : null;
    if (rect_start_coord_array != null && rect_end_coord_array != null) {
        float rect_X_start = Float.parseFloat(rect_start_coord_array[0]);
        float rect_Y_start = Float.parseFloat(rect_start_coord_array[1]);
        float rect_X_end = Float.parseFloat(rect_end_coord_array[0]);
        float rect_Y_end = Float.parseFloat(rect_end_coord_array[1]);
        structure = new Rectangle().createAndFillRectangle(0, rect_X_start, rect_X_end, spacing,
rect_Y_start, rect_Y_end);
    } else {
        System.out.println("Input of Rectangle Not received in correct format");
    }
    try {
        if (!file.exists()) {
            file.createNewFile();
        }
        outputWriter = new BufferedWriter(new FileWriter(file));
        outputWriter.write(structure);

```



```

        outputWriter.newLine();
        outputWriter.flush();
    } catch (IOException ex) {
        System.out.println("IOException while writing" + ex);
        request.setAttribute("message", "rectfunctionCall_IOException" + ex);
    }
    int lines = lineCount();
    readFile(FILENAME, request, response, context, lines);
}

}
}
bufferedReader1.close();
} catch (IOException ex) {
    System.out.println("IOException" + ex);
}
}
}

```

```

public void readFile(String fileName, HttpServletRequest request, HttpServletResponse response,
ServletContext context, int lineNumber) {
    SerialPort serialMotorPort = (SerialPort) context.getAttribute("portObject");
    SerialPort serialNozzlePort = (SerialPort) context.getAttribute("portObject1");
    // sb = (StringBuffer) context.getAttribute("message");
    // sb1 = (StringBuffer) context.getAttribute("message");
    String line = null;
    FileReader fileReader = null;
    try {
        fileReader = new FileReader(file);
    } catch (FileNotFoundException ex) {
        System.out.println("FileNotFoundException" + ex);
    }
    bufferedReader = new BufferedReader(fileReader);
    try {
        while ((line = bufferedReader.readLine()) != null) {
            try {
                System.out.println("Move segment by segment" + " " + line);
                if (!"".equalsIgnoreCase(line)) {
                    serialMotorPort.writeString(line + "\r");
                    String msg = "Move segment by segment" + " " + line;
                    sb.append(msg + "\n");
                    context.setAttribute("message", sb);
                    sb = (StringBuffer) context.getAttribute("message");
                    request.setAttribute("message", sb);
                }
            } catch (SerialPortException ex) {
                System.out.println("SerialPortException serialMotorPort writeString" + ex);
                MyServletContextListener.setFileLastLine("");
                break;
            }
        }
    }
}

```

```

}
if ("".equalsIgnoreCase(line)) {
    System.out.println("Nothing is there" + " " + line);
    //wait here till last line response is received
    String nozzledata = MyServletContextListener.getDataNozzle(false);
    System.out.println("Nozzle is off Properly" + " " + nozzledata);
    bufferedReader.close();
    MyServletContextListener.setFileLastLine(line);
    sb.append("file is finished" + "\n");
    context.setAttribute("message", sb);
    sb = (StringBuffer) context.getAttribute("message");
    request.setAttribute("message", sb);
    break;
}
String data = MyServletContextListener.getDataMotor(false);
System.out.println("motorResponse is Ok" + "" + data);
// String msg = "motorResponse is" + " " + data;
sb.append("motorResponse is ").append(data).append("\n");
context.setAttribute("message", sb);
sb = (StringBuffer) context.getAttribute("message");
request.setAttribute("message", sb);
if ("ok\n".equalsIgnoreCase(data) || "echo:busy: processing\n".equalsIgnoreCase(data)) {
    try {
        System.out.println("Head has Started");
        serialNozzlePort.writeString(PRINTGENERATEON + "\r");
        // String msg = "Move segment by segment" + " " + line;
        sb.append("Head has Started" + "\n");
        context.setAttribute("message", sb);
        sb = (StringBuffer) context.getAttribute("message");
        request.setAttribute("message", sb);
    } catch (SerialPortException ex) {
        System.out.println("SerialPortException serialNozzlePort.writeString" + ex);
        MyServletContextListener.setFileLastLine("");
        break;
    }
}
String nozzledata = MyServletContextListener.getDataNozzle(false);
sb.append("Nozzle Response is" + nozzledata + "\n");
context.setAttribute("message", sb);
sb = (StringBuffer) context.getAttribute("message");
request.setAttribute("message", sb);
if (">".contentEquals(nozzledata)) {
    try {
        System.out.println("Nozzle responded Properly,Nozzle OFF");
        serialNozzlePort.writeString(PRINTGENERATEOFF + "\r");
        sb.append("Nozzle responded Properly,Nozzle OFF" + "\n");
        context.setAttribute("message", sb);
        sb = (StringBuffer) context.getAttribute("message");
    }
}

```

```

        request.setAttribute("message", sb);
    } catch (SerialPortException ex) {
        System.out.println("SerialPortException serialNozzlePort.writeString" + ex);
        MyServletContextListener.setFileLastLine("");
        break;
    }
}

}
} catch (IOException ex) {
    System.out.println("IOException" + ex);
}
String data = MyServletContextListener.getFileLastLine();
System.out.println("file is finished" + data);
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return "Short description";
} // </editor-fold>

@Override
public void init() {
    System.out.println("Short description");
}

}

```

## Appendix F

### Design of user interface in JSP

```
<% @page contentType="text/html" pageEncoding="UTF-8"% >
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" href="simple.css">
    <script type="text/javascript" src="jquery-3.2.1.min.js"></script>
    <!--      <script type="text/javascript" src="simple.js"></script>-->

    <title>JSP Page</title>
  </head>
  <body>
    <script type="text/javascript">
      $(document).on('click', '.xHome', function (e) {
        e.preventDefault();
        $("#xHome").submit();
      });
      $(document).on('click', '.y_home', function (e) {
        e.preventDefault();
        $("#y_home").submit();
        //regionMap("y_home");
        //alert("y_home");
      });
      $(document).on('click', '.zHome', function (e) {
        e.preventDefault();
        $("#zHome").submit();
        //regionMap("zHome");
        //alert("zHome");
      });
      $(document).on('click', '.allHome', function (e) {
        e.preventDefault();
        // alert(1);
        $("#allHome").submit();
        // regionMap("all_home");
        //alert("all_home");
      });

      // y axis negative movement
      $(document).on('click', '.y_neg100', function (e) {
        e.preventDefault();
        $("#y_neg100").submit();
      });
    </script>
  </body>
</html>
```

```

    // regionMap("y_neg100");
    //alert("y_neg100");
});
$(document).on('click', '.y_neg10', function (e) {
    e.preventDefault();
    $("#y_neg10").submit();
    // regionMap("y_neg10");
    //alert("y_neg10");
});
$(document).on('click', '.y_neg1', function (e) {
    e.preventDefault();
    $("#y_neg1").submit();
    // regionMap("y_neg1");
    //alert("y_neg1");
});
$(document).on('click', '.y_negPoint1', function (e) {
    e.preventDefault();
    $("#y_negPoint1").submit();
    //regionMap("y_negPoint1");
    //alert("y_negPoint1");
});

// y axis positive movement
$(document).on('click', '.y_pos100', function (e) {
    e.preventDefault();
    $("#y_pos100").submit();
    // regionMap("y_pos100");
    //alert("y_pos100");
});
$(document).on('click', '.y_pos10', function (e) {
    e.preventDefault();
    $("#y_pos10").submit();
    // regionMap("y_pos10");
    //alert("y_pos10");
});
$(document).on('click', '.y_pos1', function (e) {
    e.preventDefault();
    $("#y_pos1").submit();
    //regionMap("y_pos1");
    //alert("y_pos1");
});
$(document).on('click', '.y_posPoint1', function (e) {
    e.preventDefault();
    $("#y_posPoint1").submit();
    //regionMap("y_posPoint1");
    //alert("y_posPoint1");
});

```

```

});

// x axis negative movement
$(document).on('click', 'x_neg100', function (e) {
    e.preventDefault();
    $("#x_neg100").submit();
    //regionMap("x_neg100");
    //alert("x_neg100");
});
$(document).on('click', 'x_neg10', function (e) {
    e.preventDefault();
    $("#x_neg10").submit();
    // regionMap("x_neg10");
    //alert("x_neg10");
});
$(document).on('click', 'x_neg1', function (e) {
    e.preventDefault();
    $("#x_neg1").submit();
    // regionMap("x_neg1");
    //alert("x_neg1");
});
$(document).on('click', 'x_negPoint1', function (e) {
    e.preventDefault();
    $("#x_negPoint1").submit();
    // regionMap("x_negPoint1");
    //alert("x_negPoint1");
});

// x axis positive movement

$(document).on('click', 'x_pos100', function (e) {
    e.preventDefault();
    $("#x_pos100").submit();
    // regionMap("x_pos100");
    //alert("x_pos100");
});
$(document).on('click', 'x_pos10', function (e) {
    e.preventDefault();
    $("#x_pos10").submit();
    //regionMap("x_pos10");
    //alert("x_pos10");
});
$(document).on('click', 'x_pos1', function (e) {
    e.preventDefault();
    $("#x_pos1").submit();
    //regionMap("x_pos1");

```

```

        //alert("x_pos1");
    });
$(document).on('click', '.x_posPoint1', function (e) {
    e.preventDefault();
    $("#x_posPoint1").submit();
    //regionMap("x_posPoint1");
    //alert("x_posPoint1");
});

// z axis positive movement
$(document).on('click', '.z_pos10', function (e) {
    e.preventDefault();
    $("#z_pos10").submit();
    // regionMap("z_pos10");
    //alert("x_pos10");
});
$(document).on('click', '.z_pos1', function (e) {
    e.preventDefault();
    $("#z_pos1").submit();
    //regionMap("z_pos1");
    //alert("x_pos1");
});
$(document).on('click', '.z_posPoint1', function (e) {
    e.preventDefault();
    $("#z_posPoint1").submit();
    // regionMap("z_posPoint1");
    //alert("x_posPoint1");
});

// z axis negative movement

$(document).on('click', '.z_neg10', function (e) {
    e.preventDefault();
    $("#z_neg10").submit();
    //regionMap("z_neg10");
    //alert("x_pos10");
});
$(document).on('click', '.z_neg1', function (e) {
    e.preventDefault();
    $("#z_neg1").submit();
    // regionMap("z_neg1");
    //alert("x_pos1");
});
$(document).on('click', '.z_negPoint1', function (e) {
    e.preventDefault();
    $("#z_negPoint1").submit();

```

```

        // regionMap("z_negPoint1");
        //alert("x_posPoint1");
    });
    $(document).ready(function () {
        var $textarea = $('#INPUT_TEXT');
        $textarea.scrollTop($textarea[0].scrollHeight);

<%

        ServletContext sc = request.getServletContext();
        sc.setAttribute("SerialPortException");

%>

        // document.getElementById('#loading').style.display = value ? 'block' : 'none';
        // var intervalID = window.setInterval(checkReady, 1000);
        //
        // function checkReady() {
        //     if (document.getElementsByTagName('body')[0] !== undefined) {
        //         window.clearInterval(intervalID);
        //         callback.call(this);
        //     }
        // }
        // }
    });

    // function show(id, value) {
    //     document.getElementById(id).style.display = value ? 'block' : 'none';
    // }
    //
    // onReady(function () {
    //     var $textarea = $('#INPUT_TEXT');
    //     show($textarea[0], true);
    //     show('loading', false);
    // });
    function eraseText() {
        document.getElementById("INPUT_TEXT").value = "";
    }

    function eraseText() {
        document.getElementById("INPUT_TEXT").value = "" }
    function yesnoCheck() {
        if (document.getElementById('dot').checked) {
            document.getElementById('dotPanel').style.display = 'block';
        } else {
            document.getElementById('dotPanel').style.display = 'none';

```



```

    }

    if (document.getElementById('line').checked) {
        document.getElementById('linePanel').style.display = 'block';
    } else {
        document.getElementById('linePanel').style.display = 'none';
    }

    if (document.getElementById('rectangle').checked) {
        document.getElementById('rectPanel').style.display = 'block';
    } else {
        document.getElementById('rectPanel').style.display = 'none';
    }

    if (document.getElementById('circle').checked) {
        document.getElementById('circlePanel').style.display = 'block';
    } else {
        document.getElementById('circlePanel').style.display = 'none';
    }
}

function singleMultiCheck() {

    if (document.getElementById('single').checked) {
        document.getElementById('drawStructure').style.display = 'block';
    } else {
        document.getElementById('drawStructure').style.display = 'none';
    }
}

$(function () {

//      $('#popuplink').click(function () {
//          $('#popup').show("slow");
//      });

    $('#popupclose').click(function () {
        $('#popup').hide("slow");
    });
});

function trueFalseCheck() {
    //nozzle 1
    if (document.getElementById('driver1').checked) {
        $('#popup').show("slow");
    }
}

```

```

        //nozzle 2
        if (document.getElementById('driver2').checked) {
            $('#popup').show("slow");
        }
    }

</script>
<!--      -->
    <div class="container" style="position: relative">
        
        
        <map name="mapname">
            <form action="NewServlet" method="post" id="xhome">
                <input type="hidden" name="param1" value="param1 Value">
                <area class="xHome" id="xHome" onfocus="blur();" onClick="xhome.submit();"
shape="poly"
coords="18,16,18,32,34,20,48,18,14,54,12,18,14,42,48,12,52,14,12,16,12,12,10,52,40,10,52,10,
50,18,26,38,22,10" href="" alt="xHome">
            </form>
            <form action="NewServlet" method="post" id="zHome">
                <input type="hidden" name="param2" value="param2 Value">
                <area class="zHome" id="zHome" onfocus="blur();" onClick="zHome.submit();"
shape="poly"
coords="226,24,228,12,230,32,234,32,234,42,234,48,208,16,198,14,224,14,238,14,238,28,238,3
8,236,48,236,54,230,48,220,42,214,26,206,20,200,18,238,20,232,52,214,28,200,14,226,14,236,4
0,224,24,236,50,196,12,204,12,234,10,216,26,224,38" href="" alt="zHome">
            </form>
            <form action="NewServlet" method="post" id="allHome">
                <input type="hidden" name="param3" value="param3 Value">
                <area class="allHome" id="allHome" onfocus="blur();"
onClick="allHome.submit();" shape="poly"
coords="16,197,20,206,30,217,42,223,54,222,16,215,14,223,12,188,24,191,36,206,40,220,46,22
3,16,226,10,203,16,202,26,201,26,197,30,209,44,219,52,229,10,225,34,229,58,225,44,215,36,21
6,20,203,12,184,20,201,20,197,12,215,18,213,30,213,34,219,58,220,18,201,18,211,38,219"
href="" alt="allHome">
            </form>

            <form action="NewServlet" method="post" id="y_home">
                <input type="hidden" name="param4" value="param4 Value">
                <area class="y_home" id="y_home" onfocus="blur();" onClick="y_home.submit();"
shape="poly"
coords="222,217,210,216,232,196,230,195,230,188,238,200,234,212,214,220,204,222,200,222,
202,212,218,226,226,226,206,220,204,214,226,212,198,222,218,223,234,224,222,198,228,188,2

```

34,190,228,228,200,227,200,224,234,220,232,203,232,188,236,224,204,220,210,211,226,196,194,224" href="" alt="y\_home">

</form>

<form action="NewServlet" method="post" id="y\_neg100">  
<input type="hidden" name="param5" value="param5Value">  
<area class="y\_neg100" id="y\_neg100" onfocus="blur();" onlick="y\_neg100.submit();" shape="poly" coords="192,204,164,214,158,218,132,228,148,218,158,220,180,220,152,224,138,226,104,224,66,210,108,210,144,210,58,204,118,212,118,232,154,230,94,226,70,218,56,206,146,212" href="" alt="y\_neg100">

</form>

<form action="NewServlet" method="post" id="y\_neg10">  
<input type="hidden" name="param6" value="param6Value">  
<area class="y\_neg10" onfocus="blur();" id="y\_neg10" onlick="y\_neg10.submit();" shape="poly" coords="82,184,92,192,102,196,114,200,92,198,76,194,84,180,136,196,156,200,166,188,128,188,166,192,176,184,122,188,140,204,150,206,160,198,178,194,94,200,108,204,74,188,70,186,68,190,76,182,102,202,138,208,94,204,92,186,106,192,110,188,78,180,144,202,146,190,164,192,172,188,122,192" href="" alt="y\_neg10">

</form>

<form action="NewServlet" method="post" id="y\_neg1">  
<input type="hidden" name="param7" value="param7Value">  
<area class="y\_neg1" onfocus="blur();" id="y\_neg1" onlick="y\_neg1.submit();" shape="poly" coords="94,170,108,172,134,180,140,174,152,162,152,172,114,180,94,174,88,170,94,162,112,164,130,172,136,170,140,166,150,166,156,166,152,174,146,178,132,184,114,186,92,176,154,172,160,170,158,174,152,180,154,160,128,174,114,172,106,164,112,166,96,166,140,182" href="" alt="y\_neg1">

</form>

<form action="NewServlet" method="post" id="y\_negPoint1">  
<input type="hidden" name="param8" value="param8Value">  
<area class="y\_negPoint1" onfocus="blur();" id="y\_negPoint1" onlick="y\_negPoint1.submit();" shape="poly" coords="120,150,124,144,134,148,108,158,120,160,126,156,138,156,132,148,130,142,112,142,108,154,104,154,128,156,142,150,134,140,118,154,114,150,114,146,118,144,122,144,130,164,140,152,144,154,102,152,134,142,136,156,116,156,126,158,132,152" href="" alt="y\_negPoint1">

</form>

<form action="NewServlet" method="post" id="y\_pos100">  
<input type="hidden" name="param9" value="param9Value">  
<area class="y\_pos100" onfocus="blur();" id="y\_pos100" shape="poly" onlick="y\_pos100.submit();" coords="64,30,102,22,108,14,124,14,144,16,168,26,176,30,64,34,60,34,60,42,64,44,68,42,86,20

```

,90,18,94,16,96,16,102,14,96,20,150,14,164,20,170,24,188,32,186,38,158,28,170,36,148,26,104,
24,74,24,122,10,130,10,136,8,106,10" href="" alt="y_pos100">
</form>
<form action="NewServlet" method="post" id="y_pos10">
  <input type="hidden" name="param10" value="param10Value">
  <area class="y_pos10" onfocus="blur();" id="y_pos10" shape="poly"
onClick="y_pos10.submit();"
coords="82,48,94,46,102,46,106,38,78,48,80,56,88,56,84,64,72,54,96,52,116,42,136,42,158,40,
164,50,164,56,172,50,142,40,150,52,166,56,162,58,176,50,142,34,86,40,74,48,104,46,98,52,114
,48,136,46,142,48,148,60,138,52,154,58,68,50,136,48,134,52,148,44,104,34,146,40,142,38,152,
36,160,46,160,56" href="" alt="y_pos10">
</form>
<form action="NewServlet" method="post" id="y_pos1">
  <input type="hidden" name="param11" value="param11Value">
  <area class="y_pos1" onfocus="blur();" id="y_pos1" shape="poly"
onClick="y_pos1.submit();"
coords="98,72,106,68,118,64,132,64,136,72,146,76,150,66,146,80,88,66,102,58,104,58,98,80,1
16,58,140,58,142,60,144,68,158,70,154,78,106,56,92,70,96,76,106,76,136,62,150,72,126,74,110
,76,130,68,132,58,148,64,156,68,160,68,88,70,92,78,106,62,128,58,100,70,120,68,138,64,136,6
8,130,70,138,72" href="" alt="y_pos1">
</form>
<form action="NewServlet" method="post" id="y_posPoint1">
  <input type="hidden" name="param12" value="param12Value">
  <area class="y_posPoint1" onfocus="blur();" id="y_posPoint1" shape="poly"
onClick="y_posPoint1.submit();"
coords="124,92,114,88,126,86,122,100,130,90,138,86,108,86,110,90,116,96,134,96,138,80,126,
78,114,78,106,84,106,90,114,98,124,98,132,98,140,88,142,86,116,84,116,80,126,82,126,92,118,
98,130,100,116,94,106,86,110,82,130,82,132,90,146,88,102,84,108,92,116,98,118,102,120,88,1
28,78,132,80" href="" alt="y_posPoint1">
</form>
<form action="NewServlet" method="post" id="x_pos100">
  <input type="hidden" name="param13" value="param13Value">
  <area class="x_pos100" onfocus="blur();" id="x_pos100" shape="poly"
onClick="x_pos100.submit();"
coords="210,174,212,166,222,152,206,188,212,190,198,184,208,192,210,186,200,178,218,176,
222,162,206,172,212,158,218,148,218,144,222,136,224,164,216,184,230,148,230,140,226,134,2
32,124,232,102,232,138,222,102,218,86,210,70,218,140,224,92,216,76,212,62,210,58,222,122,2
30,96,228,82,220,66,212,54,202,60,226,98,222,72,214,64,222,96,226,142,222,164,216,178,230,
108,230,100" href="" alt="x_pos100">
</form>
<form action="NewServlet" method="post" id="x_pos10">
  <input type="hidden" name="param14" value="param14Value">
  <area class="x_pos10" onfocus="blur();" id="x_pos10" shape="poly"
onClick="x_pos10.submit();"
coords="198,154,188,162,194,142,202,130,200,104,196,86,188,76,196,68,204,96,186,82,194,74
,198,94,202,102,208,104,202,156,190,170,192,174,180,166,196,174,204,154,206,144,198,146,1

```

```

94,160,186,164,206,138,208,152,210,142,198,138,190,160,194,130,196,108,192,100,190,98,194
,88,208,96,202,80,194,64,188,170,182,162,196,166,208,144,210,136" href="" alt="x_pos10">
</form>
<form action="NewServlet" method="post" id="x_pos1">
  <input type="hidden" name="param15" value="param15Value">
  <area class="x_pos1" onfocus="blur();" id="x_pos1" shape="poly"
onClick="x_pos1.submit();"
coords="176,96,174,112,174,116,176,122,176,134,172,144,168,152,174,152,182,128,184,98,17
6,88,166,96,166,112,164,140,172,154,186,138,176,158,184,130,186,122,188,118,186,104,180,9
0,178,100,180,114,172,106,170,98,172,96,172,114,172,136,168,138,164,148,172,156,182,140,1
82,124,170,124,166,126,168,128,186,136,186,130,180,100,176,84" href="" alt="x_pos1">
</form>
<form action="NewServlet" method="post" id="x_posPoint1">
  <input type="hidden" name="param16" value="param16Value">
  <area class="x_posPoint1" onfocus="blur();" id="x_posPoint1"
onClick="x_posPoint1.submit();" shape="poly"
coords="154,120,154,112,150,122,148,112,158,104,164,136,154,128,148,128,156,136,154,138,
158,144,144,124,144,112,156,102,162,108,162,122,156,124,146,136,150,118,156,106,156,120,1
56,126" href="" alt="x_posPoint1">
</form>
<form action="NewServlet" method="post" id="x_neg100">
  <input type="hidden" name="param17" value="param17Value">
  <area class="x_neg100" onfocus="blur();" id="x_neg100"
onClick="x_neg100.submit();" shape="poly"
coords="26,152,36,176,18,132,16,108,24,86,40,174,38,184,44,184,40,192,46,178,26,166,42,184
,34,154,30,136,22,146,22,156,26,170,36,180,34,70,42,54,28,80,24,80,24,114,16,112,16,128,42,5
6,36,54,40,60,48,62,28,96,30,62,20,90,14,108,16,134,18,146,24,124,16,100,28,74,38,60,40,52,4
0,66,20,118,32,170,32,178,46,182" href="" alt="x_neg100">
</form>
<form action="NewServlet" method="post" id="x_neg10">
  <input type="hidden" name="param18" value="param18Value">
  <area class="x_neg10" onfocus="blur();" id="x_neg10"
onClick="x_neg10.submit();" shape="poly"
coords="54,82,52,96,46,116,48,142,50,162,54,164,38,148,52,174,64,152,58,144,64,156,56,152,
44,96,48,78,60,70,60,84,66,84,54,96,54,104,42,136,40,142,44,152,40,102,44,86,46,74,56,66,64,
74,56,154,60,170,64,164,60,172,54,178,62,164" href="" alt="x_neg10">
</form>
<form action="NewServlet" method="post" id="x_neg1">
  <input type="hidden" name="param19" value="param19Value">
  <area class="x_neg1" onfocus="blur();" id="x_neg1" shape="poly"
onClick="x_neg1.submit();"
coords="70,98,68,134,72,140,72,146,80,142,76,154,72,154,64,134,64,116,66,106,74,96,78,88,7
8,106,74,146,84,150,76,156,70,152,66,140,60,120,62,102,70,86,84,98,76,120,74,128,72,144,72,
156,78,134,82,144,70,158,62,136,76,84,86,92" href="" alt="x_neg1">
</form>
<form action="NewServlet" method="post" id="x_negPoint1">

```

```

        <input type="hidden" name="param20" value="param20Value">
        <area          class="x_negPoint1"          onfocus="blur();"          id="x_negPoint1"
onClick="x_negPoint1.submit();"          shape="poly"
coords="92,112,90,126,102,118,88,108,90,96,104,112,84,120,88,132,88,140,98,126,96,140,104,
130,84,108,86,114,86,124,88,134,104,122" href="" alt="x_negPoint1">
    </form>
</map>

<map name="mapname1">
    <form action="NewServlet" method="post" id="z_pos10">
        <input type="hidden" name="param21" value="param21Value">
        <area          class="z_pos10"          onfocus="blur();"          id="z_pos10"          shape="poly"
onClick="z_pos10.submit();"
coords="14,44,14,44,28,48,28,48,38,50,38,50,42,42,42,42,52,42,52,22,56,22,56,14,58,36,60,
36,60,42,58,42,58,40,40,40,40,42,58,42,58,44,44,44,44,46,42,46,42,44,58,44,58,12,56,12,56,12,
48,12,48,12,40,12,40,20,40,20,40,48,42,48,42,44,54,44,54,30,62,30,62,8,60,8,60,24,50,24,50,26,
48,26,48,28,42,28,42,32,34,32,34,52,38,52,38,12,52,12,52,30,44,30,44,42,60,42,60" href=""
alt="z_pos10">
    </form>
    <form action="NewServlet" method="post" id="z_pos1">
        <input type="hidden" name="param22" value="param22Value">
        <area          class="z_pos1"          onfocus="blur();"          id="z_pos1"          shape="poly"
onClick="z_pos1.submit();"
coords="20,74,20,74,40,78,40,78,18,78,18,78,18,84,18,84,12,76,12,76,12,66,12,66,26,66,26,66,
26,66,38,68,38,68,46,72,46,72,44,82,44,82,16,86,16,86,38,88,38,88,40,80,40,80,24,76,24,76,20,
72,20,72,16,72,16,72,16,64,16,64,16,68,16,68,14,76,14,76,14,86,14,86,42,88,42,88,44,70,44,70,
40,74,40,74,44,70,44,70,46,82,46,82,42,66,42,66,44,66,44,66,44,82,44,82,32,76,32,76,16,78,16,
78,14,64,14,64,38,74,38,74,44,74,44,74,46,86,44,68,44,68" href="" alt="z_pos1">
    </form>
    <form action="NewServlet" method="post" id="z_posPoint1">
        <input type="hidden" name="param23" value="param23Value">
        <area          class="z_posPoint1"          onfocus="blur();"          id="z_posPoint1"
onClick="z_posPoint1.submit();"          shape="poly"
coords="18,94,18,94,28,104,28,104,42,102,42,102,20,106,20,106,14,102,14,102,44,94,44,94,42,
106,42,106,16,96,16,96,12,94,12,94,14,106,14,106,38,106,38,106,50,100,50,100,32,94,32,94,42,
104,42,104,26,106,26,106,14,108,14,108,42,112,42,112,42,98,42,98,40,94,40,94,16,94,16,94"
href="" alt="z_posPoint1">
    </form>
    <form action="NewServlet" method="post" id="z_neg10">
        <input type="hidden" name="param24" value="param24Value">
        <area          class="z_neg10"          onfocus="blur();"          id="z_neg10"
onClick="z_neg10.submit();"          shape="poly"
coords="24,182,24,182,34,184,34,184,10,184,10,184,38,192,38,192,20,196,20,196,40,194,40,19
4,38,184,38,184,40,172,22,176,22,176,14,176,14,176,34,186,34,186,32,180,32,180,44,178,44,17
8,44,200,44,200,16,192,16,192,16,180,16,180,16,194,16,194,14,192,14,192,12,178,12,178,40,17

```

```

8,40,178,24,192,24,192,32,182,32,182,16,188,16,188,14,198,14,198,44,196,44,196,44,178,44,17
8,40,184,40,184" href="" alt="z_neg10">
    </form>
    <form action="NewServlet" method="post" id="z_neg1">
        <input type="hidden" name="param25" value="param25Value">
        <area class="z_neg1" onfocus="blur();" id="z_neg1" onClick="z_neg1.submit();"
shape="poly"
coords="14,152,14,152,20,156,20,156,30,158,30,158,30,158,40,154,40,154,16,162,16,162,34,16
6,34,166,36,160,36,160,36,160,46,152,46,152,48,164,48,164,38,164,38,164,14,166,14,166,34,17
0,34,170,18,150,18,150,26,148,26,148,30,150,30,150,44,154,44,154,36,168,36,168,42,168,42,16
8,24,156,24,156,14,162,14,162,12,170,12,170,26,156,26,156,34,150,34,150,12,156,12,156,14,15
4,14,154,40,154,40,154,42,152,42,152,18,160,18,160,14,154,14,154,42,152,42,152,40,160,40,16
0,28,164,28,164,42,166,42,166,20,168,20,168,12,158,12,158" href="" alt="z_neg1">
    </form>
    <form action="NewServlet" method="post" id="z_negPoint1">
        <input type="hidden" name="param26" value="param26Value">
        <area class="z_negPoint1" onfocus="blur();" id="z_negPoint1"
onClick="z_negPoint1.submit();" shape="poly"
coords="20,132,20,132,32,136,32,136,20,142,20,142,18,124,18,124,40,126,40,126,40,132,40,13
2,40,140,40,140,48,132,48,132,38,140,38,140,10,142,10,142,12,134,12,134,16,126,16,126,36,14
4,36,144,44,130,44,130,44,144,44,144,24,144,24,144,18,134,18,134,28,126,28,126,34,130,34,13
0,42,130,42,130,20,138,20,138,32,136,32,136,28,130,28,130,40,138,40,138,16,138,16,138,12,13
0,12,130,28,128,28,128,42,124,42,124" href="" alt="z_negPoint1">
    </form>
</map>
<div class="fixed">
    <h3> Choose File to Upload and Print</h3>
    <form action="NewServlet" method="post" enctype="multipart/form-data">
        <input type="file" name="file" />
        <input type="submit" value="Print" />
    </form>
</div>

<div>
    <textarea id="INPUT_TEXT" name="INPUT_TEXT" style="font-size: 13pt;"
rows="10" cols="40" class="fill-form-font">${message}</textarea>
    <form action="NewServlet" method="post">
        <input type="text" name="instruct" class="fill-form-font"/>
        <input type="submit" name="sendInstruct" value="Send"/>
        <input type="button" value="Clear" onclick="javascript:eraseText();">
    </form>
</div>
<div>
    <form action="NewServlet" method="post">
        <input type="submit" value="Pulse ON" name="pulseOn">

```

```

        <input type="submit" value="Pulse OFF" name="pulseOff">
    </form>
</div>

</div>

<div id="drawStructure">
    <fieldset style="width:350px">
        <legend>Please select any of the option !!</legend>
        <label><input type="radio" onclick="javascript:yesnoCheck();" name="toggle"
id="dot"><span>Dot</span></label>
        <label><input type="radio" onclick="javascript:yesnoCheck();" name="toggle"
id="line"><span>Line</span></label>
        <label><input type="radio" onclick="javascript:yesnoCheck();" name="toggle"
id="retangle"><span>Rectangle</span></label>
        <label><input type="radio" onclick="javascript:yesnoCheck();" name="toggle"
id="circle"><span>Circle</span></label>
    </fieldset>

    <div id="dotPanel" style="display:none">
        <form action="NewServlet" method="post">
            <fieldset>
                <legend>Enter value(in mm)!</legend>
                Start Cord(X,Y): <input type="float" name ="start_coord"/>
                <input type="submit" name="dotSubmit" value="Submit"/>
            </fieldset>
        </form>
    </div>

    <div id="linePanel" style="display:none">
        <form action="NewServlet" method="post">
            <fieldset>
                <legend>Enter value(in mm)!</legend>
                Start Cord(X,Y): <input type="float" name ="start_coord"/>
                End Cord(X,Y): <input type="float" name ="end_coord"/>
                Spacing <input type="float" name ="space"/>
                <input type="submit" name="lineSubmit" value="Submit"/>
            </fieldset>
        </form>
    </div>

    <div id="rectPanel" style="display:none">
        <form action="NewServlet" method="post">
            <fieldset>
                <legend>Enter value(in mm)!</legend>

```



```

        Start Cord(X,Y): <input type="float" name ="start_coord"/>
        End Cord(X,Y): <input type="float" name ="end_coord"/>
        Spacing <input type="float" name ="space"/>
        <input type="submit" name="rectSubmit" value="Submit"/>
    </fieldset>
</form>
</div>
<div id="circlePanel" style="display:none">
    <form action="NewServlet" method="post">
        <fieldset>
            <legend>Enter Coordinates value(in mm)!</legend>
            Center(X,Y): <input type="float" name ="circle_centre"/>
            Radius: <input type="float" name ="radius"/>
            Spacing: <input type="float" name ="space"/>
            <input type="radio" name="direction" value="Clockwise" checked> Clockwise
            <input type="radio" name="direction" value="Anti-Clockwise"> Anti-Clockwise
            <input type="submit" name="circleSubmit" value="Submit"/>
        </fieldset>
    </form>
</div>
</div>

```