Automatizing nano-processing stage and optical delay line

Sanjay Kapoor

A dissertation submitted for the partial fulfillment of BS-MS dual degree in Science

Department of Physical Sciences
Indian Institute of Science Education and Research
Mohali

April 2019

Certificate of Examination

This is to certify that the dissertation titled "**Automatizing nano-processing stage and optical delay line**" submitted by Sanjay Kapoor (Reg. No. MS14099) for the partial fulfillment of BS-MS dual degree programme of the institute, has been examined by the thesis committee duly appointed by the institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Samir Kumar Biswas

Dr. Abhishek Chaudhuri	Dr. Samir Kumar Biswas	Dr. Kamal P. Singh (Supervisor)

Dated: April 25, 2019

Declaration

The work presented in this dissertation has been carried out by me under guidance of Dr. Kamal P. Singh at the Indian Institute of Science Education and Research Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have detailed in the bibliography.

Sanjay Kapoor
(Candidate)

Dated: April 25, 2019

In my capacity as the supervisor of the candidate's project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Dr. Kamal P. Singh
(Supervisor)

Acknowledgements

I thank my supervisor Dr. Kamal P. Singh for allowing me to work with him. It was a great experience to have him as a supervisor and mentor to learn from him.

I am thankful to Dr. Mehra Singh Sidhu for helpful discussions about the projects.
I would like to thank all the lab members of femtosecond laser lab Amit Kumar, Akansha Tyagi, Biswjit Panda, Komal Chaudhary, Pawan Suthar, Pooja Munjal, Ritika Dagar, Shivali Sokhi, Sunil Dahiya, Varun Ranade, and Vishavdeep Vashisht for helping me whenever needed.

I want to thank my family for their love, support, and freedom.

I thank all of my friends Anjani Gupta, Ankuj Kumar, Dharmendra Kumar, Harsh Verma, Jagmeet Singh, Manpreet Singh, Pawan Suthar, Raman Choudhary, Shashank Prakash, Shivam Mishra, and Vishal Kumar Sharma.

A special thanks to Rishabh and Vasudev for reviewing this dissertation.

List of Figures

List of Tables

Contents

Dedicated to my grandfather.

Abstract

An all-reflective dispersion-free optical delay line was implemented with custom made mechanical parts. A custom *LabVIEW* program was written to automate the scanning of the delay steps with a resolution of 27 *as* over a range of 533 *fs*. The delay line was characterized for collinearity, delay steps, stability, and time zero. The stability of the delay line was found to be 57 *as* over a distance of 107 *cm* for about 40 *s*.

A motorized high-speed XY microscope stage was automated in *LabVIEW* to move on given (x, y) coordinate using both X and Y motors simultaneously. A high-speed electronic shutter was interfaced with the same *LabVIEW* program. A GUI *Python3* program was written to draw arbitrary patterns on an image of the region of interest.

Chapter 1

Mechanical design and characterization of automated optical delay line

1.1 Introduction

Optical delay lines (ODL) are similar to two beam interferometers. These are used to introduce the desired time delay in one beam by controlling the optical path length. ODLs of various designs are used in many application and optical devices [1]. ODLs are used in ultrafast pulse measurements with autocorrelator [2], optical coherence tomography [3], IR-IR and IR-XUV pump-probe experiments [4].
An all-reflective dispersion free ODL has been already developed and being used in autocorrelator for ultra-short pulse measurements in femtosecond laser lab at IISER Mohali. We have implemented the same idea of the ODL for IR-IR pump-probe experiments with fs pulses. We aimed to develop an ultra stable delay line with high precision, high repeatability, and with enough scanning range to capture ultra-fast phenomena. Piezoelectric stack actuator (PZT) for with strain gauge sensor is used to introduce delay. For quick, precise, and good repeatability of the delay steps we automated the scanning using *LabVIEW*.

1.2 Dispersion-free all-reflective optical delay line design

The main idea is to split the profile of the laser beam with a knife edge prism mirror shown in the Figure 1.1. The splitted beams travel through two arms implemented by two mirrors (V-block). Using an another knife edge prism mirror, the splitted beams are directed in the propagation direction. The spacing between splitted beams can be adjusted with second knife edge prism mirror. Figure 1.1 shows the design of the delay line based on knife edge prism mirrors. The idea is to split the laser beam profile with knife edge into two parts. The splitted beams then travels through two arms , using another knife edge prism mirror the splitted beams are directed in the propagation direction and the spacing between splitted beam can adjusted with placement of the second knife edge prism mirror.

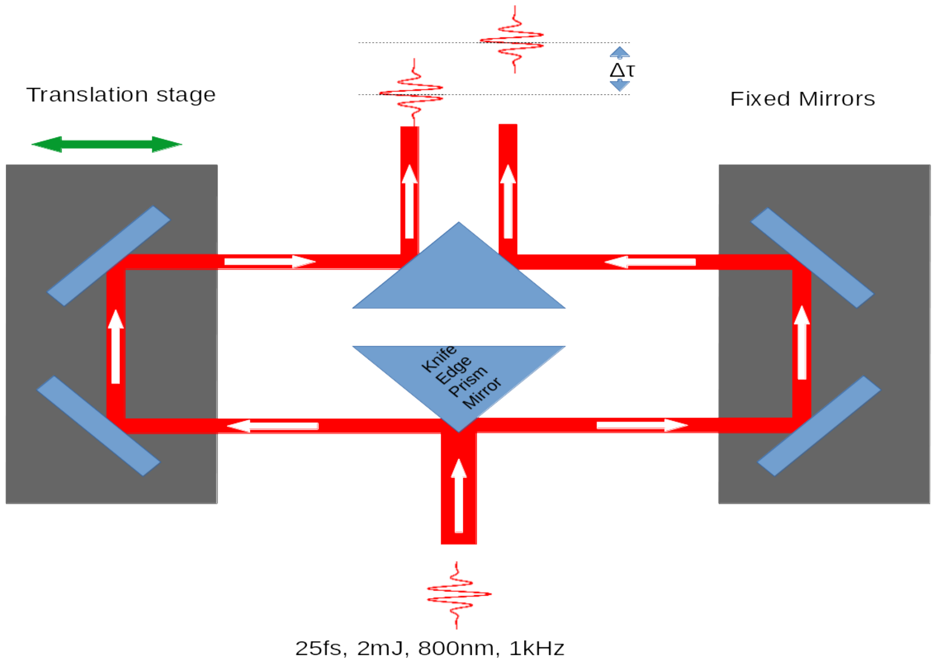


FIGURE 1.1: All-reflective dispersion-free optical delay line based on knife edge prism mirrors

V-block on the right side is mounted on manual linear translation stage for coarse adjustment. V-block on the left side of the knife edge is mounted on PZT which is used to introduce the optical delay.

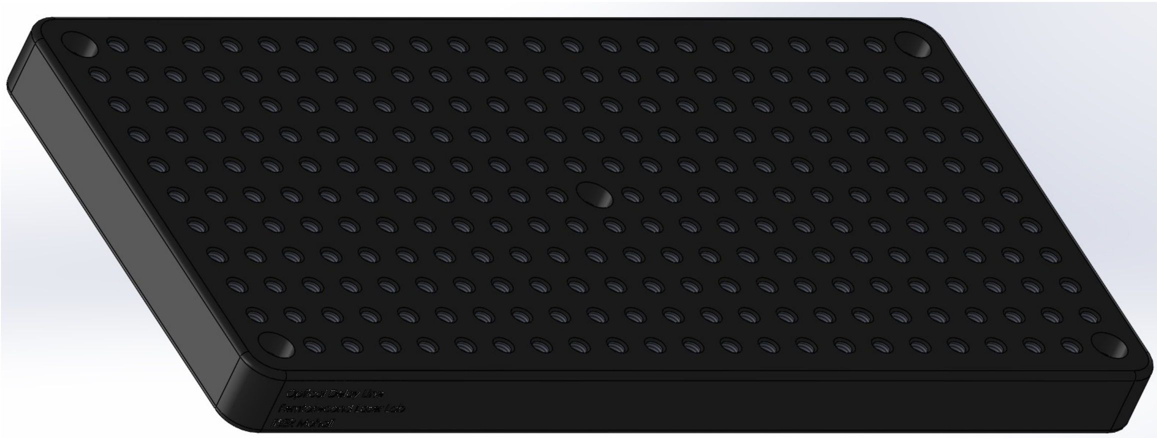
If the left arm of the delay line is longer than the right arm by Δd then corresponding time delay $\Delta\tau$ is given by

$$\Delta\tau = \frac{2\Delta d}{c}, \text{ where c is the speed of light.} \tag{1.1}$$

The factor of 2 in Equation 1.1 is because of the geometry of the delay line. The pulse traveling though the left arm has to go the twice the extra displacement of the left V-block. The PZT has a maximum stroke of 80 μm in closed-loop operation. Using Equation 1.1 the full scanning range of PZT corresponds to a time delay of 533.3 fs.

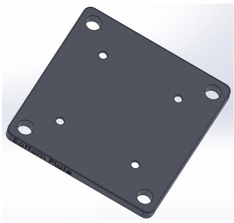
1.3 Mechanical design of the delay line

The beam height of fs laser from the optical bench is fixed (≈202 mm). So we had to design our own mechanical parts to meet this constraint. Designed mechanical parts in *Solid Works* are shown in Figure 1.2 with their dimensions. *Solid Works* assembly of the ODL is shown in Figure 1.3.

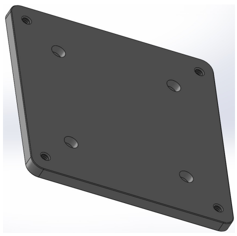


(A) Optical bench $304.8 \times 152.4 \times 22\ mm^3$ with $12.7\ mm\ M6$ grid

(B) Adaptor for PZT and V-block ($90 \times 88 \times 5\ mm^3$)

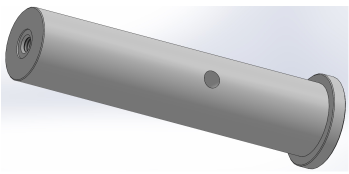


(C) Adaptor for ODL bench and PZT ($62 \times 62 \times 4.7\ mm^3$)

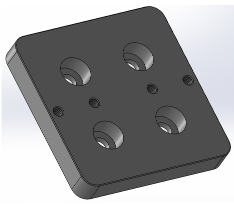


(D) Adaptor for LNR25M/M and V-block ($90 \times 88 \times 5\ mm^3$)

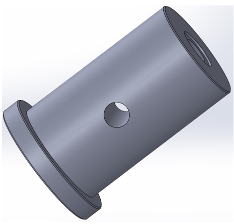
(E) Adaptor for LNR25M/M and Figure 1.2d ($60 \times 60 \times 18.7\ mm^3$)



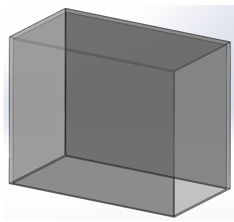
(F) Pedestal post $\varnothing 25.4mm$ ($115.30mm$)



(G) Prism mounting plate ($60 \times 60 \times 9\ mm^3$)

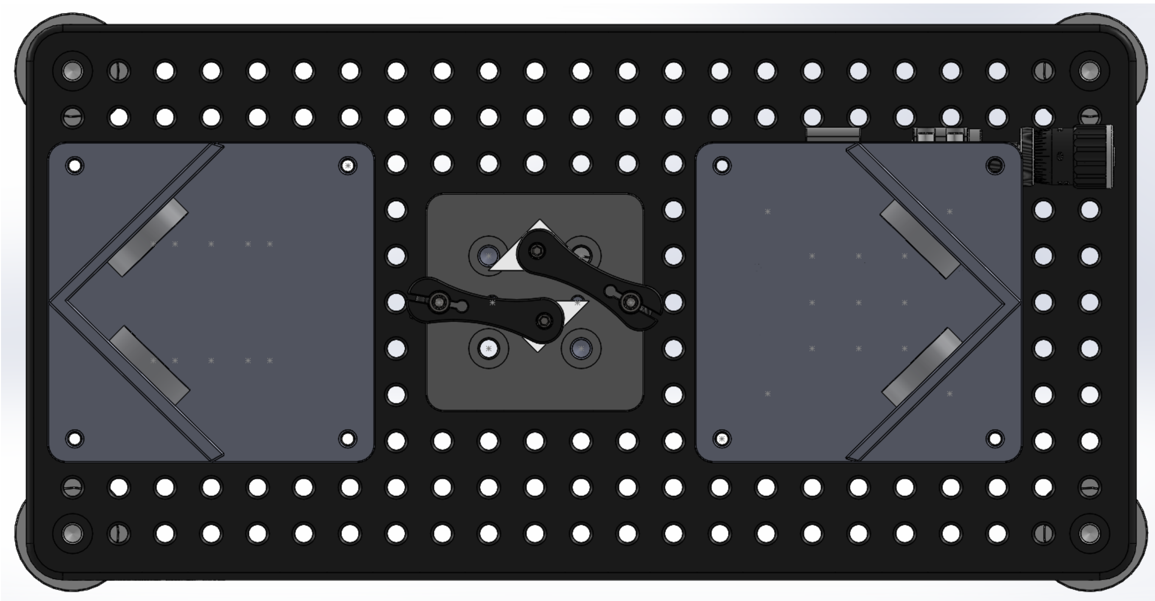


(H) Pedestal post $\varnothing 25.4mm$ ($45.70\ mm$)

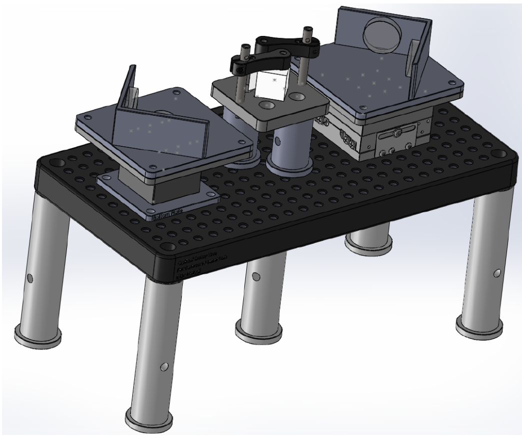


(I) Acrylic cover for ODL ($333 \times 265 \times 180\ mm^3$)

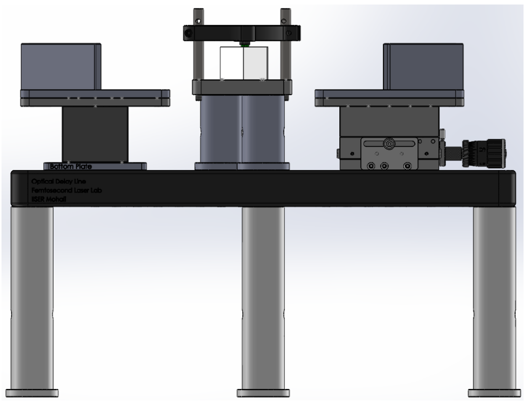
FIGURE 1.2: Custom made mechanical parts of the delay line (*Solid Workds* design)



(A) Top view



(B) Isometric view



(C) Side view

FIGURE 1.3: *Solid works* assembly of the delay line

1.4 Automatizing the delay line

A custom-designed software based on *LabVIEW* is made to automate the delay line. It can communicate to the PZT controller through its system commands. A scan with discrete delay steps over a user-defined range can be initiated from this software. Figure 1.4 shows the overview of the scan tab. For more details of the program see Appendix C.

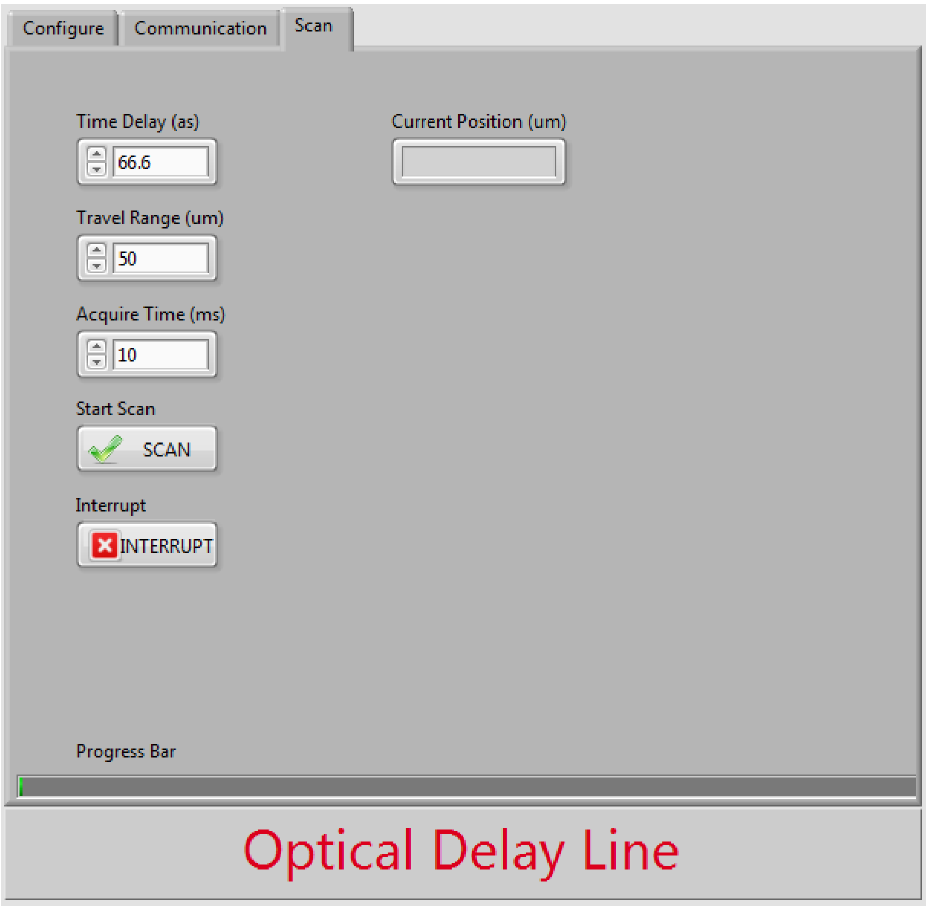


FIGURE 1.4: The scan tab of the *LabVIEW* program of optical delay line. `Delay Time (as)` is optical delay step required in *as*. `Travel Range` is the scanning range in μm. `Acquire Time (ms)` is the time for which PZT stays at a position during scan. `SCAN` button initiates the optical delay scan with steps of `Delay Time`. `Progress Bar` shows the scanning progress.

Figure 1.5 shows the schematic of the automated ODL. The setup consists of ODL, PZT controller, and a PC with *LabVIEW* installed.

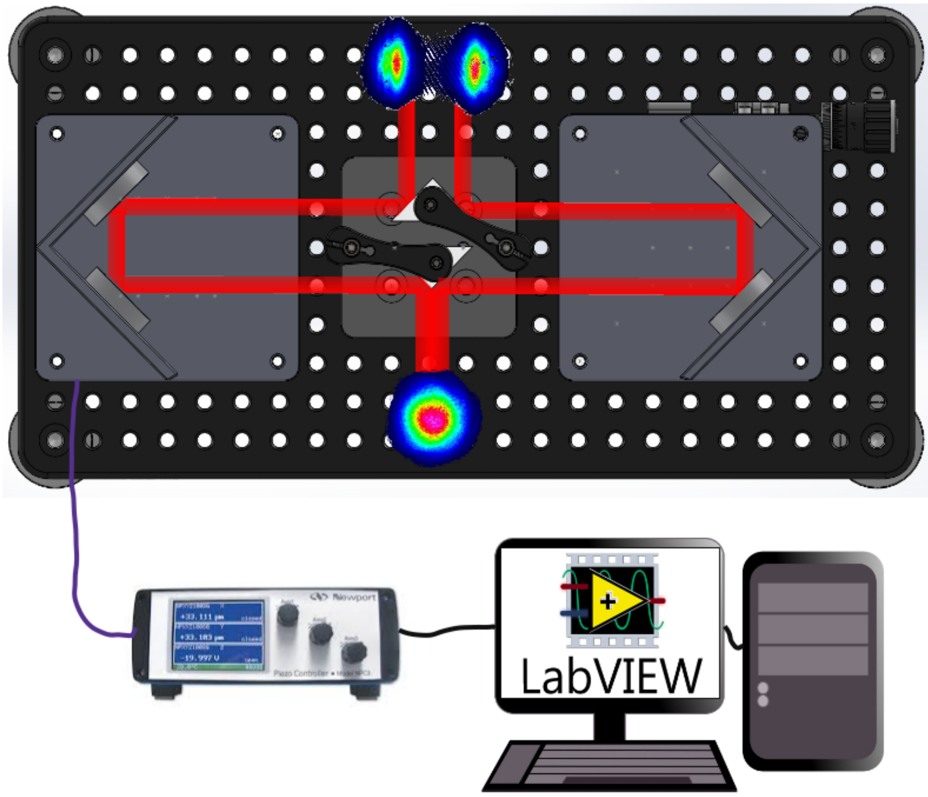


FIGURE 1.5: Schematic of the automated ODL

1.5 Characterization of the optical delay line

After assembling and automating the delay line, we aligned the setup and characterized the ODL for the following parameters:

1. Collinearity: Two emerging beams are parallel to each other and the optical bench.
2. Calibration of the delay steps.
3. Time zero calibration: To make the two arm lengths equal (within a few microns).
4. Stability of the system over a long period.

Height alignment, collinearity, strain gauge cross verification, and stability were easily tested with continuous wave HeNe laser ($\lambda = 632.8\ nm$) but time zero cannot be found

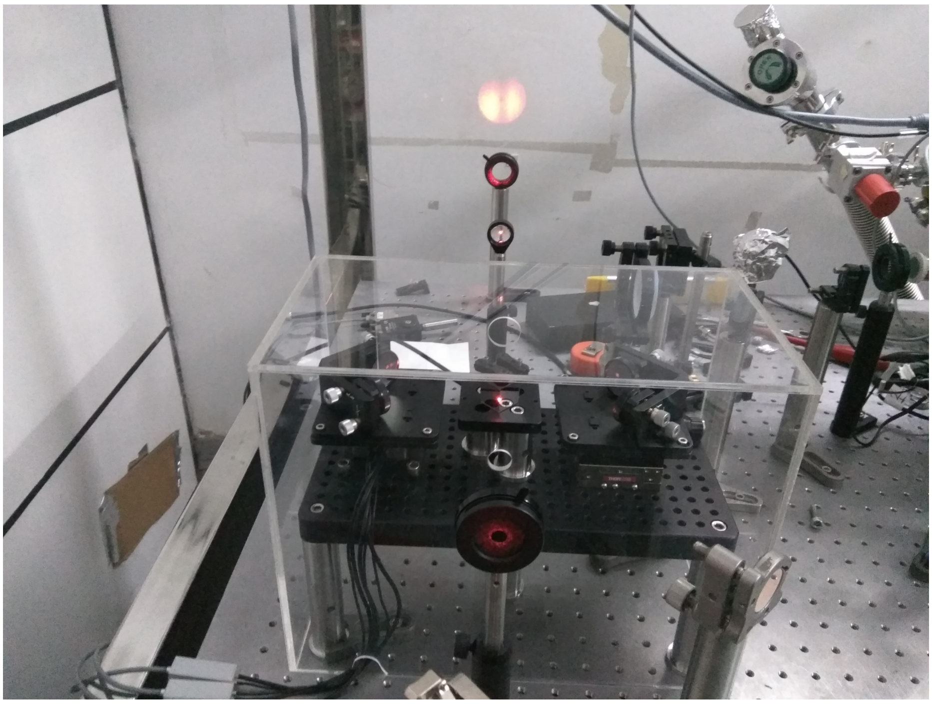


FIGURE 1.6: Actual setup of ODL (with HeNe laser).

using HeNe laser because of its long coherence length. To find time zero we used fs pulses, and estimated how close the arm lengths were.

1.5.1 Alignment of the delay line

Steps involved in alignment of the ODL

1. Used two irises of the same height ($\approx 202\ mm$) to align the HeNe laser beam.
2. Placed the delay line in between two irises as shown in Figure 1.6.
3. Adjusted the splitting knife edge prism mirror to split the beam into roughly two equal half, such that both splitted beams hit the mirrors at the center.
4. Fastened the CF-175 clamps on the pedestal posts to lock the delay line on the optical bench.
5. Placed the second knife edge prism mirror such that two splitted beams are as close possible.
6. Adjusted the kinematic V-block mirrors to make the beam collinear and pass through the second iris.

1.5.2 Cross verification of the delay steps

According to the specification of the PZT in closed loop the resolution is 4 nm, see the Table C.1. The displacement of PZT is internally measured with a strain gauge. To check the reliability of the strain gauge, we used the HeNe laser and obtained the interference pattern by deliberately misaligning the setup to get better fringe contrast. Figure 1.7 shows the interference pattern with HeNe laser. One fringe shift (from bright to dark and bright fringe again) corresponds to $\lambda/2 \approx 317\ nm$ and by counting the fringe shifts, the information about change in path length can be obtained. Figure 1.8 is the plot of photodiode (PD) signal vs PZT position which shows periodicity (Λ) of the signal (one complete fringe shift) agrees with $\lambda/2$ within 1 nm.

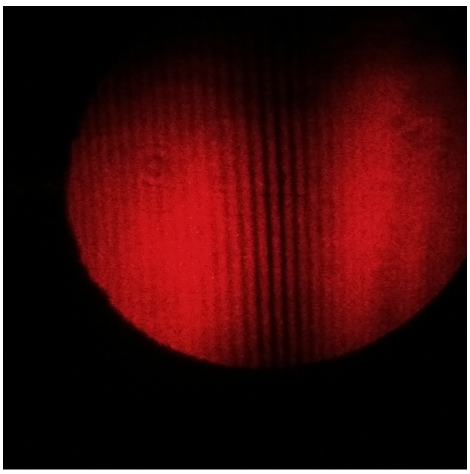


FIGURE 1.7: Fringes with HeNe laser.

Figure 1.9 shows the PD signal for a

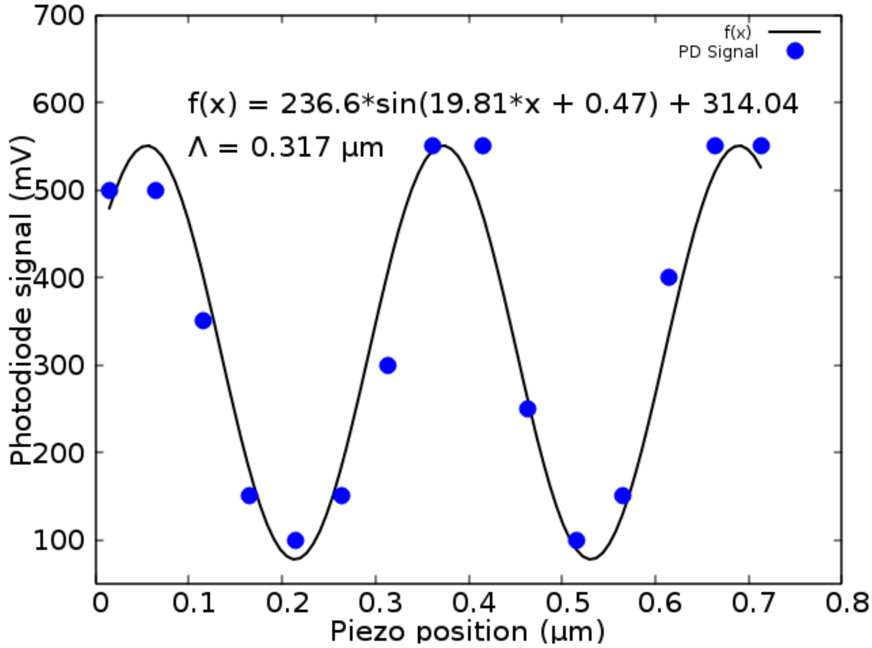
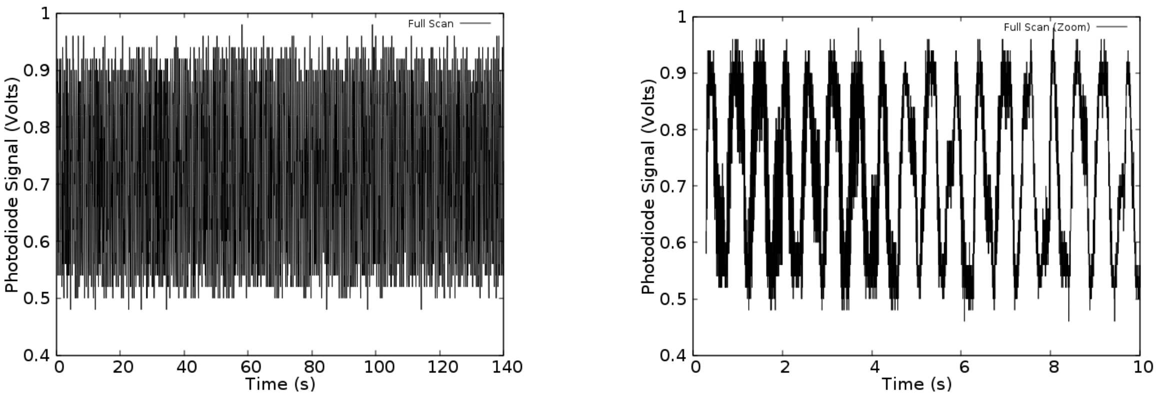


FIGURE 1.8: Cross verification of strain gauge with HeNe laser ($\lambda = 633\ nm$).



(A) Full scan of the piezo (0 to 80 μm).

(B) Inset plot of the full scan.

FIGURE 1.9: PD signal for full scan.

full scan range (0 – 533 fs or 0 – 80 μm)
with 1 fs delay steps and 10 ms acquire time. Oscillations due to fringe shifts are clearly visible in Figure 1.9b verifying the working of automated scanning with the *LabVIEW* program.

1.5.2.1 Comparison of set value and strain gauge reading

An offset of about 14.6 nm in the strain gauge reading displayed by the PZT controller and set value (command through the software) was observed (Figure 1.10).

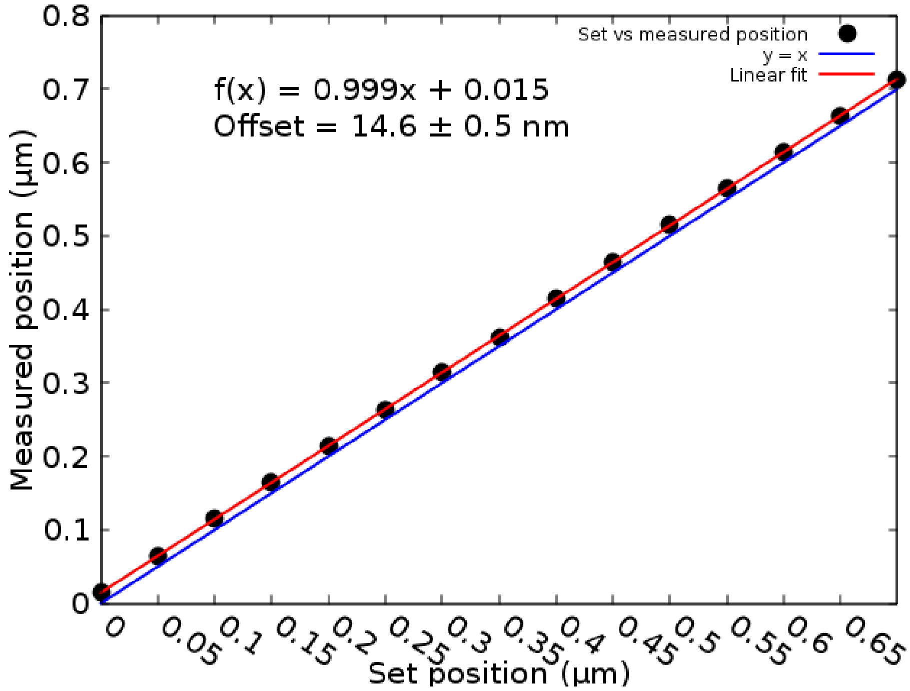


FIGURE 1.10: Comparison between set value and strain gauge reading. Blue line is the y = x line, on which strain gauge readings were expected. Red line is the linear fit through the strain gauge readings.

This offset is not a problem for delay steps because of its consistency across the scanning range.

1.5.3 Stability of the delay line

We have obtained fringes with a HeNe laser and placed the PD in the interference pattern. Fringe width was made nearly equal to the active area of the PD using a diverging lens. Scanned few fringes and then fixed the set point in somewhere middle of the PD voltage and acquire the data for $\approx 50\ s$ with the oscilloscope. Plot of the PD signal vs time is shown in Figure 1.11 red line in the plot shows the average PD signal after the set point and light yellow background show the $\pm\sigma$ about the average.

$$\text{Minimum voltage, } V_{min} = 0.114 \text{ V} \tag{1.2}$$

$$\text{Maximum voltage, } V_{max} = 0.574 \text{ V} \tag{1.3}$$

$$\Delta V = V_{max} - V_{min} = 0.46\text{V} \tag{1.4}$$

$$\text{Standard deviation in the data after the set point, } \sigma = 0.05 \text{ V} \tag{1.5}$$

ΔV corresponds to $\lambda/4 \approx 158.2\ nm$ (half fringe shift).

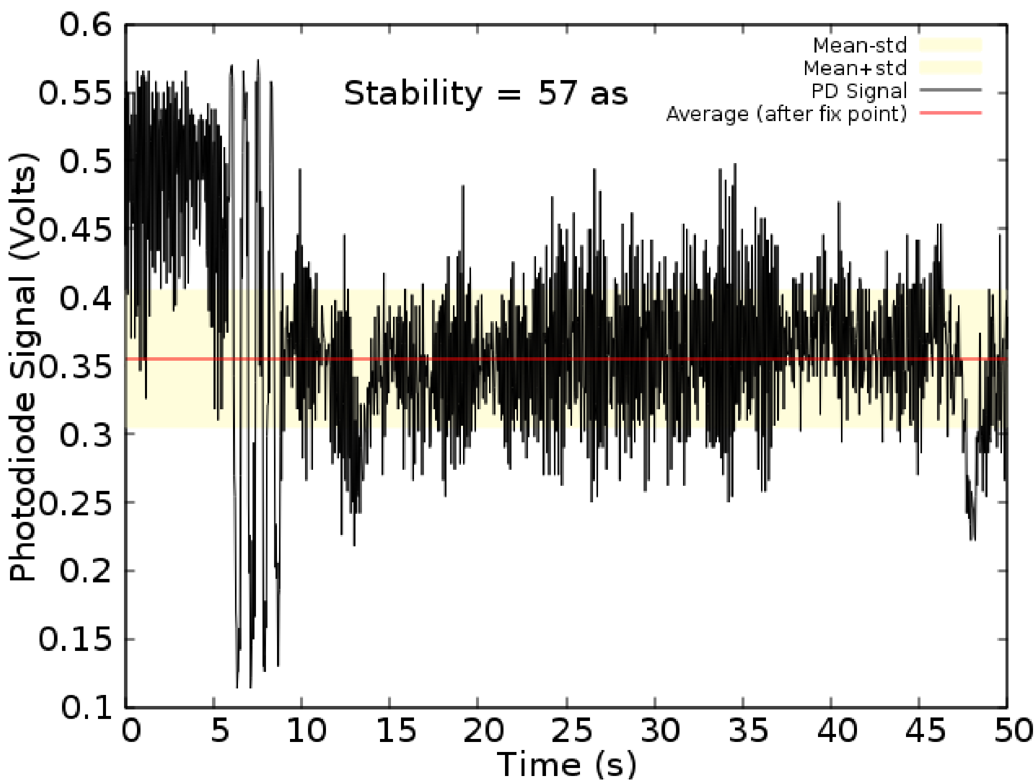


FIGURE 1.11: Stability of the delay line is 57 *as* or 17 *nm* fluctuation in path lengh over a distance of $\approx 107\ cm$ for $\approx 40\ s$.

$$\text{Fluctuation in path length} = \sigma \times \frac{\lambda}{4\Delta V} = 17.2\ nm \tag{1.6}$$

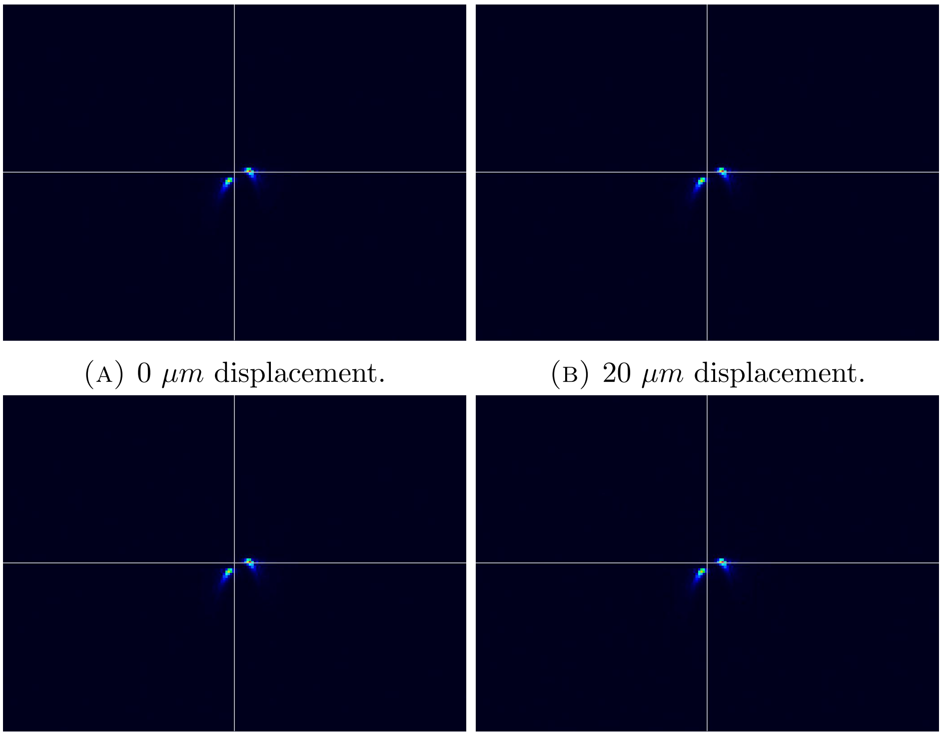
Now stability in time is calculated dividing the fluctuation in path length by the speed of light

$$\text{Stability} = 57.3\ as \tag{1.7}$$

Stability of the delay line is 57 as over a distance of $\approx$ 107 cm for roughly 40 seconds.

1.5.4 Collinearity

To check the collinearity, we used a converging lens to focus the splitted beams and then placed beam profiler at the focus. For different PZT displacements the beam profile was captured at the focus. Figure 1.12 shows the captured images of the beam profile at the focus for different displacements of the PZT.



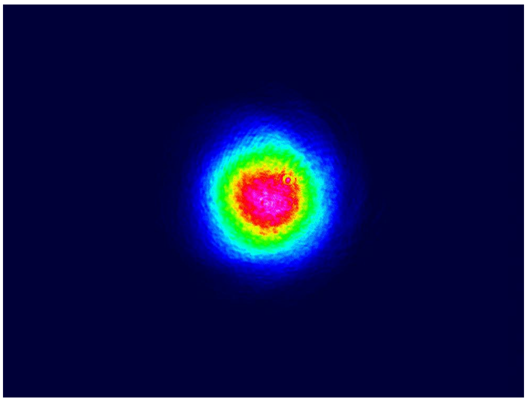
(A) 0 μm displacement. (B) 20 μm displacement.

(C) 40 μm displacement. (D) 80 μm displacement.

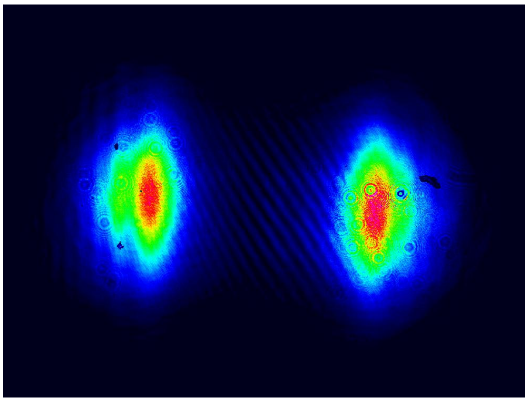
FIGURE 1.12: Beam profile images at focus for different displacements of PZT.

There is no noticeable change in the position of captured profiles indicating very good collinearity.

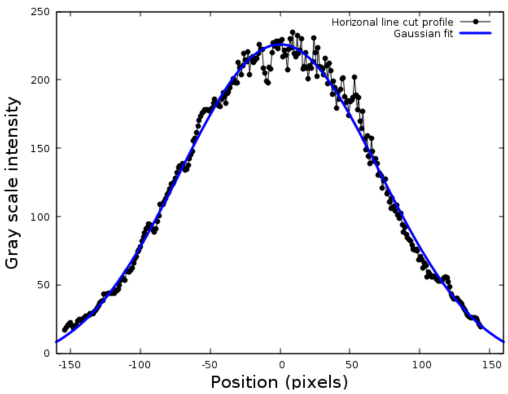
Figure 1.13 shows the profile of the input HeNe laser beam and profile after splitting with knife edge prism mirror. In Figure 1.13b interference pattern is due to a small overlap of the splitted beam caused by the divergence of the laser beam.



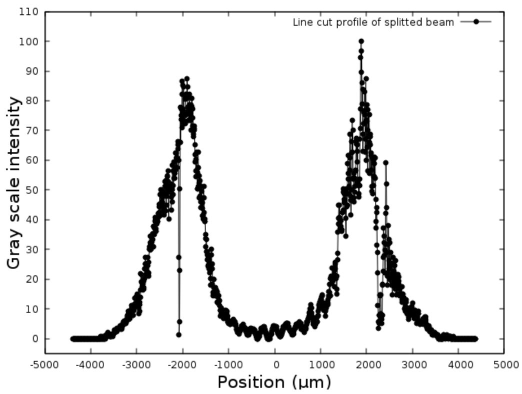
(A) HeNe laser beam profile at input of the delay line.



(B) HeNe laser beam profile at output of the delay line (after splitting with knife edge prisms).



(C) HeNe line cut (along horizontal) profile.



(D) Line cut (along horizontal) profile of the splitted beam.

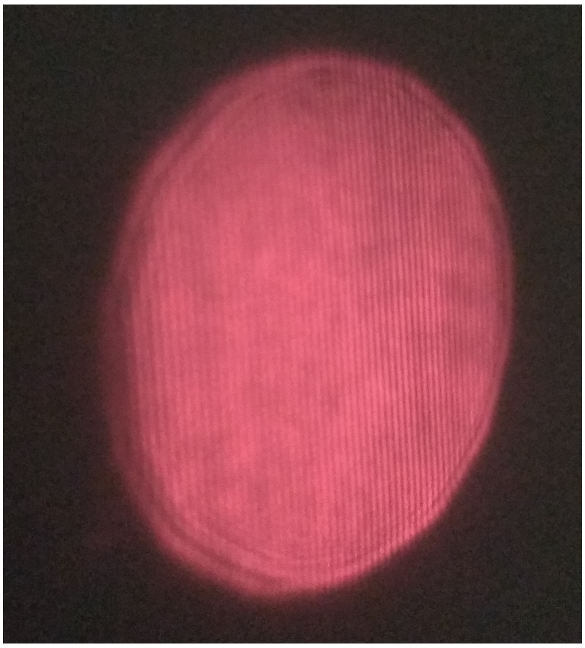
FIGURE 1.13: HeNe laser beam profile before and after splitting with knife edge mirror.

There is an unexpected dip in both the splitted parts of the laser beam visible in line cut profile shown in Figure 1.13d.

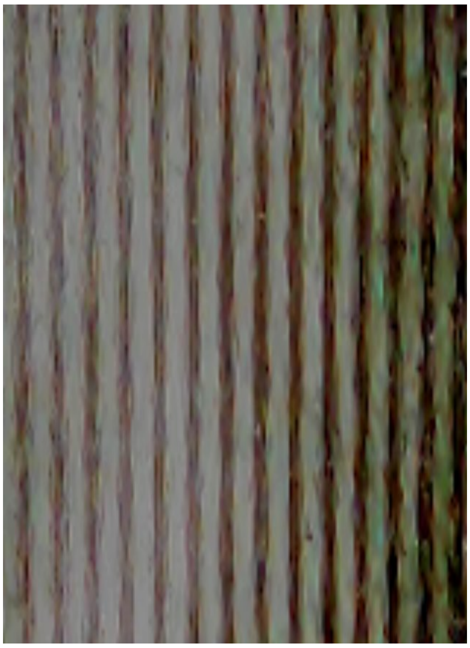
1.5.5 Time zero calibration

Time zero in optical delay line means when both arm lengths are exactly equal i.e. there is no time delay between the pulses. We have incorporated a linear translation stage (25 mm travel range) with 12.7 μm resolution.

Time zero cannot be found with HeNe laser because of its high coherence length ($>$ 20 cm). We used ultrashort (fs) pulses to find the time zero. Figure 1.14a shows the interference pattern with fs pulses. We recorded a video of the fringes as PZT scanned for full range (0 to 80 μm). Figure 1.15 shows the contrast of a fixed pixel in the video as PZT was scanning. Change in the fringe contrast is visible in Figure 1.15 and we see that primary interference pattern appears in a range of 15 μm which means arm lengths are same within 15 μm.



(A) Interference pattern with fs pulses



(B) Close-up of interference pattern with fs pulses captured by WebCam.

FIGURE 1.14: Time zero fringes.

1.6 Summary

An all-reflective dispersion-free optical delay line was implemented with custom made mechanical parts. A custom *LabVIEW* program was written to automate the scanning

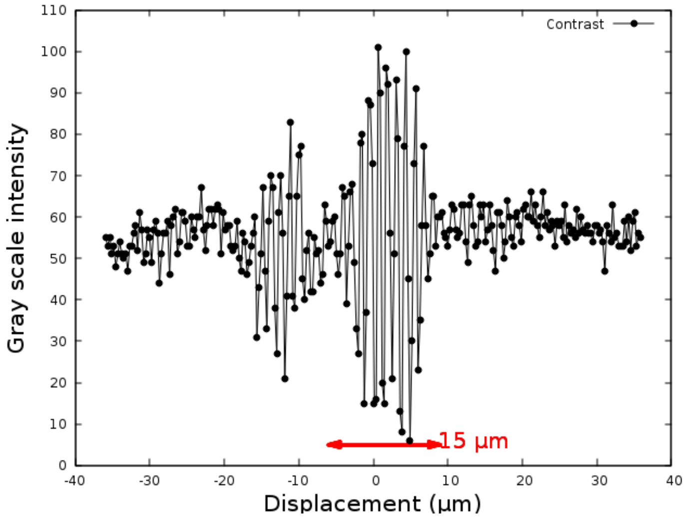


FIGURE 1.15: Gray scale value at a fixed point in the video as PZT was scanning.

of the delay steps with a resolution of 27 *as* over a range of 533 *fs*. The delay line was characterized for collinearity, delay steps, stability, and time zero. The stability of the delay line was found to be 57 *as* over a distance of 107 *cm* for about 40 *s*.

Chapter 2

Automatizing the laser processing stage

2.1 Introduction

A 2 mJ, 25 fs pulse has peak power of

$$P_{peak} = \frac{2\text{mJ}}{25\text{fs}} = 800\text{GW} \tag{2.1}$$

due to such high peak powers fs laser processing facilitates smooth cutting with minimal thermal (or collateral) damage. Here is comparison between holes drilled in a 100 μm thick stainless steel foil using 3.3 ns and 200 fs pulses. In both cases fluences just above the ablation threshold and 10,000 pulses have been used [5]. Time scale of

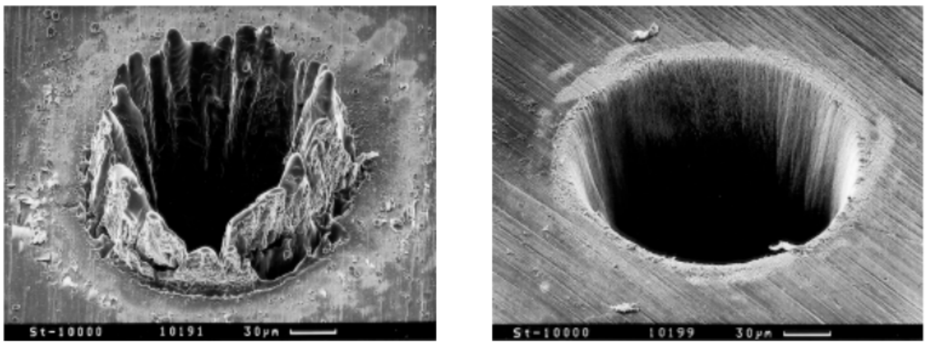


Figure 2.1: Comparison between ns and fs laser processing (adapted from [5])

ultra-short laser processing is shown in the Figure 2.2, energy diposition happens on time scale of pulse duration (fs) while ablation can last upto microsceond regime [5].

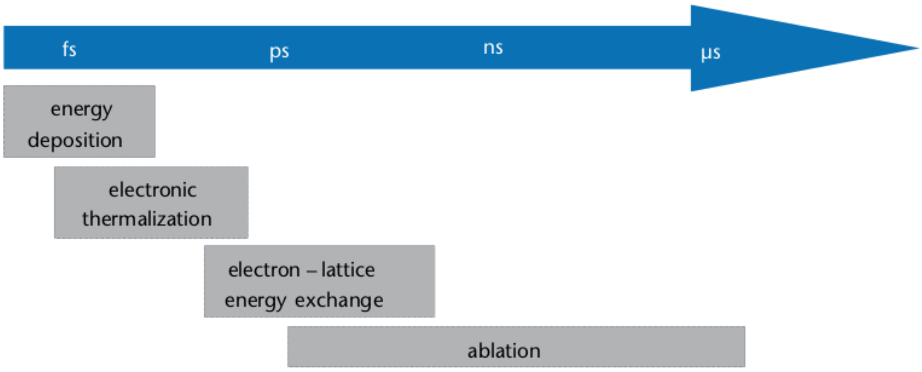


FIGURE 2.2: Schematic of time scale for ultra short pulse ablation (adapted from [5])

In this project we aimed at automating the fs processing setup in the Femtosecond Laser Lab at IISER Mohali.

2.2 Automated femtosecond laser processing setup

Femtosecond laser processing setup is shown in Figure 2.3. The setup consists of

1. Ti:Sapphire laser which generate 2 mJ, 25 fs pulses with centeral wavelengh about 800 nm at repetition rate of 1 kHz.
2. High speed electronic shutter with 6 mm aperture.
3. Neutral density filter (NDF).
4. Microscope with 4x, 10x, 20x, 40x, and 100x objectives; integrated with CCD camera (5 MP), and motorized x-y-stage.
5. Stage controller
6. Shutter controller
7. Computer with LabVIEW and Python3 installed.

The duration of laser illumination was controlled by controlling the open duration of the shutter. The microscope stage, camera, shutter were controlled by custom-made *LabVIEW* based software which is described in Appendix B. The computer that we used had Intel core i5, 8 GB RAM and on-board integrated video card.

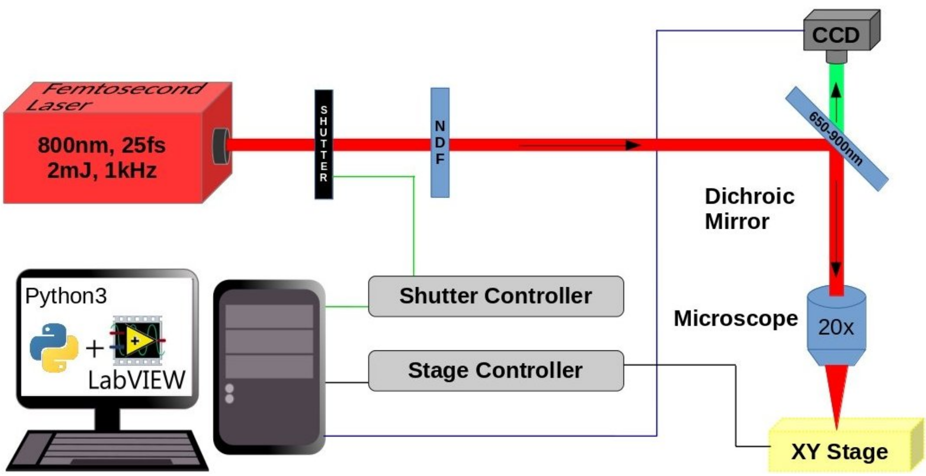


FIGURE 2.3: Schematic of the automated femtosecond laser processing setup

The software is written in two separate programming languages. *LabVIEW* based custom program `FSLProcessor.vi` controls several hardwares components and `FSLPencil.py` determine the laser illumination locations through the graphical software made in *Python3*.

2.3 Flow diagram of the software

The software consists of two separate programs written in *Python3* (`FSLPencil.py`) and *LabVIEW* (`FSLProcessor.vi`) given in Appendix A, Appendix B respectively. `FSLPencil.py` is GUI based on `Tkinter` [6] for *Python3*. With this program one can draw arbitrary patterns on an image of the ROI and writes the coordinates of drawn pattern into a text file. Figure 2.4 shows the flow diagram diagram of the combined software.

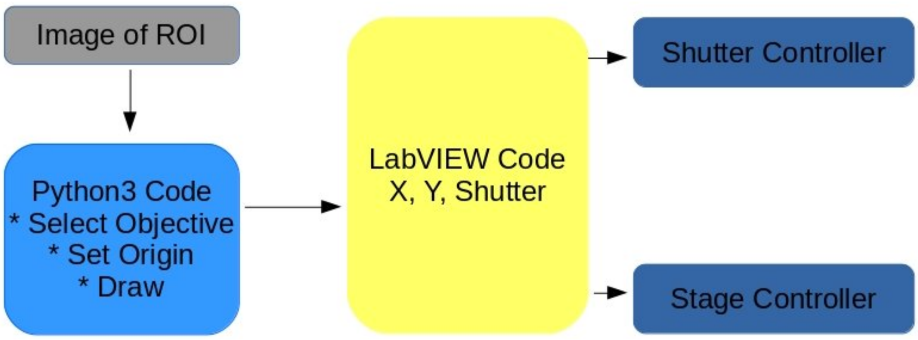
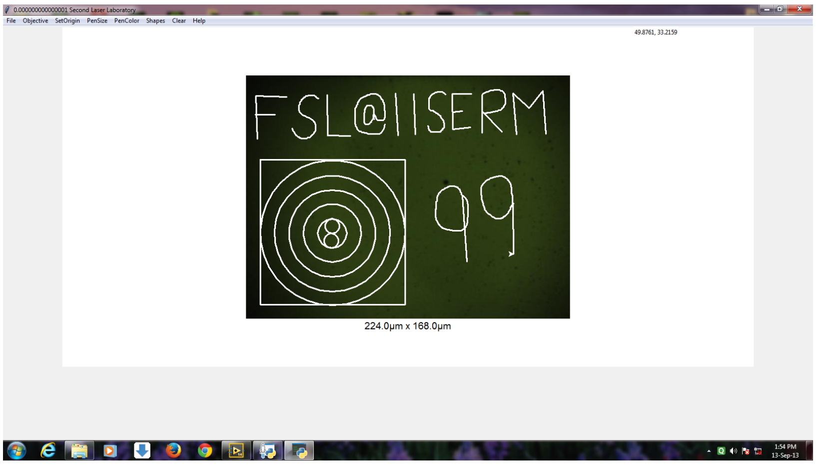


FIGURE 2.4: Flow diagram of the software

`FSLProcessor.vi` take the text file as input which has three columns (x, y, shutter state) and moves the stage to (x, y) position with the shutter stage (on/off).

2.4 Results

fs laser drawn pattern on glass slide using 20x objective.



(A) A pattern on ROI drawn in `FSLPencil.py`

(B) fs laser drawn pattern on glass slide (Objective: 20x, Spot size: 4.4μm, Average power: 3mW, Processing time: 19min 20sec)

FIGURE 2.5: (A) Pattern drawn in computer and (B) fs laser drawn pattern on glass slide.

2.5 Summary

A motorized high-speed XY microscope stage was automated in *LabVIEW* to move on given (x, y) coordinate using both X and Y motors simultaneously. A high-speed electronic shutter was interfaced with the same *LabVIEW* program. A GUI *Python3* program was written to draw arbitrary patterns on an image of the region of interest.

Appendix A

Python program to draw arbitrary patterns on the ROI

This is the python3 [7] code `FSLPencil.py` to draw arbitary pattern on the ROI image taken by the camera in the setup. The program is graphical user interface based on `Tkinter` [6] and `Turtle` graphics [8].

A.1 FSLPencil.py

```
from turtle import*
from tkinter import*
from tkinter import messagebox
from tkinter import simpledialog
from tkinter import filedialog
from tkinter.colorchooser import*
from PIL import Image

root = Tk()
root.title("0.000000000000001 Second Laser Laboratory")
root.resizable(width=True, height=True)

MainFrame = Frame(root, borderwidth=1, padx=5, pady=5)
```

```python
MainFrame.pack()

canvas = Canvas(master = root, width=1366, height=670)
canvas.pack()

var = StringVar()
xycor = Label(root, textvariable=var, bg="white", fg="black")
xycor.place(x=1245, y=2)

t = RawTurtle(canvas)
t.speed(0)
screen = t.getscreen()
t.penup()
catFile = open("target.txt", "w")
tclick = "right"
PreviousEvent = "jump"
width, height = 640, 480      #Dimensions of the image
origin = (55, 37.5)           #Origin of the region of interest
PixelLen = 0.15               #Length of a pixel in mm (depends of the objective)
def quit():
    root.destroy()
def openfile():
    filename = filedialog.askopenfilename()
    Image.open(str(filename)).convert("RGB").save(str(filename)[:-4]+".gif")
    screen.bgpic(str(filename)[:-4]+".gif")
def label_scale():
    t.clear()
    t.goto(0, -265)
    t.pendown()
    t.write("%.1f m x %.1f m"%(640*PixelLen*1000, 480*PixelLen*1000), align="center", font=("Arial", 14, "normal"))
    t.penup()
```

```
    t.goto(0, 0)
def transform(coord):
    global PixelLen, origin
    x, y = origin[0]-coord[0]*PixelLen, origin[1]-coord[1]*
  PixelLen
    return (x, y)
def forx():
    global PixelLen
    PixelLen = 0.00176
    label_scale()
def tenx():
    global PixelLen
    PixelLen = 0.000705
    label_scale()
def twentyx():
    global PixelLen
    PixelLen = 0.00035
    label_scale()
def fortyx():
    global PixelLen
    PixelLen = 0.000177
    label_scale()
def hundredx():
    global PixelLen
    PixelLen = 7.6e-5
    label_scale()
def set_origin():
    global origin
    ox = simpledialog.askfloat("Set origin", "Enter x
  coordinate")
    oy = simpledialog.askfloat("Set origin", "Enter y
  coordinate")
    if ox == None or oy == None or ox == 0.0 or oy == 0.0:
        pass
    else:
```

```
        origin = (ox, oy)
        catFile.write("%.4f\t%.4f\t0\n"%(ox, oy))
def clear():
    t.clear()
def drawcircle():
    radius = simpledialog.askstring("Circle", "Enter radius of the circle (mm)")
    if radius == None or radius == '':
        pass
    else:
        i = (t.xcor(), t.ycor())
        i = transform(i)
        catFile.write("%.4f\t%.4f\t2\n"%(i[0], i[1]))
        catFile.write("%.4f\t%.4f\t1\n"%(i[0], i[1]))
        t.begin_poly()
        t.pendown()
        radius = float(radius)/PixelLen
        t.circle(radius)
        t.penup()
        t.end_poly()
        coords = t.get_poly()
        for i in coords:
            i = transform(i)
            catFile.write("%.4f\t%.4f\t2\n"%(i[0], i[1]))
        i = (t.xcor(), t.ycor())
        i = transform(i)
        catFile.write("%.4f\t%.4f\t0\n"%(i[0], i[1]))
def drawpoly():
    n = simpledialog.askstring("Regular Polygon", "Enter the number of sides")
    l = simpledialog.askstring("Regular Polygon", "Enter the length of side (mm)")
    if n == None or l == None or n == '' or l == '':
        pass
    else:
```

```
        i = (t.xcor(), t.ycor())
        i = transform(i)
        catFile.write("%.4f\t%.4f\t2\n"%(i[0], i[1]))
        catFile.write("%.4f\t%.4f\t1\n"%(i[0], i[1]))
        t.begin_poly()
        t.pendown()
        l = float(l)/PixelLen
        for i in range(int(n)):
            t.forward(l)
            t.left(360/int(n))
        t.penup()
        t.end_poly()
        coords = t.get_poly()
        for i in coords:
            i = transform(i)
            catFile.write("%.4f\t%.4f\t2\n"%(i[0], i[1]))
        i = (t.xcor(), t.ycor())
        i = transform(i)
        catFile.write("%.4f\t%.4f\t0\n"%(i[0], i[1]))
def helpindex():
    messagebox.showinfo("Help Index", "Sorry! No content found.")
def about():
    messagebox.showinfo("About...", "Author: Sanjay Kapoor\nemail: sanjaykapoor@protonmail.com")
def help():
    messagebox.showinfo("Help?", "Sorry! No content found.")
def pensize1():
    t.pensize(1)
def pensize2():
    t.pensize(2)
def pensize3():
    t.pensize(3)
def pensize4():
    t.pensize(4)
```

```
def pensize5():
    t.pensize(5)
def getcolor():
    color = askcolor()
    if color[1] == None:
        pass
    else:
        t.color(color[1], color[1])
def coordinates(event):
    global var
    i = (683-event.x, event.y-335)
    i = transform(i)
    xycoor = "%.4f, %.4f"%(i[0], i[1])
    var.set(xycoor)
def line(x, y):
    global tclick
    click = "left"
    i = transform(t.position())
    t.pendown()
    t.goto(x, y)
    j = transform(t.position())
    if tclick == click:
        catFile.write("%.4f\t%.4f\t2\n"%(j[0], j[1]))
    else:
        catFile.write("%.4f\t%.4f\t2\n"%(i[0], i[1]))
        catFile.write("%.4f\t%.4f\t1\n"%(i[0], i[1]))
        catFile.write("%.4f\t%.4f\t2\n"%(j[0], j[1]))
    tclick = click
def freehand(x, y):
    global PreviousEvent
    t.pendown()
    t.goto(x, y)
    i = transform(t.position())
    catFile.write("%.4f\t%.4f\t2\n"%(i[0], i[1]))
    PreviousEvent = "freehand"
```

```python
def jump(x, y):
    global tclick, PreviousEvent
    click = "right"
    i = transform(t.position())
    t.penup()
    t.goto(x, y)
    j = transform(t.position())
    if tclick != click or PreviousEvent == "freehand":
        catFile.write("%.4f\t%.4f\t0\n"%(i[0], i[1]))
    tclick = click
    PreviousEvent = "jump"
screen.onclick(line)
screen.onclick(jump, btn=3)
t.ondrag(freehand)

menubar = Menu(root)
FileMenu = Menu(menubar, tearoff=0)
FileMenu.add_command(label="Open", command=openfile)
FileMenu.add_separator()
FileMenu.add_command(label="Exit", command=quit)
menubar.add_cascade(label="File", menu=FileMenu)
ObjectiveLens = Menu(menubar, tearoff=0)
ObjectiveLens.add_radiobutton(label="4x", command=forx)
ObjectiveLens.add_radiobutton(label="10x", command=tenx)
ObjectiveLens.add_radiobutton(label="20x", command=twentyx)
ObjectiveLens.add_radiobutton(label="40x", command=fortyx)
ObjectiveLens.add_radiobutton(label="100x", command=hundredx)
menubar.add_cascade(label="Objective", menu=ObjectiveLens)
SetOrigin = Menu(menubar, tearoff=0)
SetOrigin.add_command(label="SetOrigin", command=set_origin)
menubar.add_cascade(label="SetOrigin", menu=SetOrigin)
PenSize = Menu(menubar, tearoff=0)
PenSize.add_radiobutton(label="1", command=pensize1)
PenSize.add_radiobutton(label="2", command=pensize2)
PenSize.add_radiobutton(label="3", command=pensize3)
```

```
PenSize.add_radiobutton(label="4", command=pensize4)
PenSize.add_radiobutton(label="5", command=pensize5)
menubar.add_cascade(label="PenSize", menu=PenSize)
PenColor = Menu(menubar, tearoff=0)
PenColor.add_command(label="Choose Color", command=getcolor)
menubar.add_cascade(label="PenColor", menu=PenColor)
Shapes = Menu(menubar, tearoff=0)
Shapes.add_command(label="Circle", command=drawcircle)
Shapes.add_command(label="Polygon", command=drawpoly)
menubar.add_cascade(label="Shapes", menu=Shapes)
Clear = Menu(menubar, tearoff=0)
Clear.add_command(label="Clear", command=clear)
menubar.add_cascade(label="Clear", menu=Clear)
Help = Menu(menubar, tearoff=0)
Help.add_command(label="Help Index", command=helpindex)
Help.add_command(label="About...", command=about)
Help.add_command(label="Help?", command=help)
menubar.add_cascade(label="Help", menu=Help)

root.config(menu=menubar)

def refresh():
    screen.tracer(100000)
    canvas.after(1, refresh)
refresh()
canvas.bind('<Motion>', coordinates)
root.mainloop()
i = t.position()
i = transform(i)
catFile.write("%.4f\t%.4f\t0\n"%(i[0], i[1]))          # close the shutter at last point
catFile.write("%.4f\t%.4f\t2\n"%(origin[0], origin[1])) #go back to the origin
catFile.close()
#Sanjay Kapoor; ms14099; 16.07.2018
```

Appendix B

Custom *LabVIEW* program to automate fs processing

B.1 Thorlabs MLS203-1 high speed motorized stage

The motorized stage is driven by a brush-less motor DC servo controller BBD202. The

Parameter	Value
Travel	$110mm \times 75mm$
Max speed	$250mm/s$
Acceleration	$2000mm/s^2$
Min incremental movement	$0.1\mu m$
Home location accuracy	$0.25\mu m$
Max load	$1kg$
Setting time within $1\mu m$	$0.1s$
Setting time within $0.1\mu m$	$0.6sec$

TABLE B.1: Stage specifications

motorized stage (`MLS203-1`) specifications are given in the Table B.1. The stage can be controlled by a joystick from thorlabs `MJC001`, computer program from thorlabs `APT User` which controls one motor at a time or custom program made in *LabVIEW*. *LabVIEW* is intrument interfacing software [9]. Following are the commands of BBD202 used to made the part of *LabVIEW* program `FSLProcessor.vi`

1. `HWSerialNum`

2. `StartCtrl`
3. `SetJogStepSize`
4. `SetVelParams`
5. `SetAbsMovePos`
6. `MoveAbsoluteEx`
7. `GetPosition`
8. `StopCtrl`

The *LabVIEW* program take absolute x-y coordinates (mm) and moves the stage to the specified coordinates.

B.2 Uniblitz shutter driver VMMD-3

`VMMD-3` is a three channel shutter driver it can control three `LS6S2ZM1-100` shutters simultaneously. But we need only one shutter to stop the laser beam. The shutter is integrated with the setup through `RS232` communication within same program `FSLProcessor.vi`, following are the serial port settings and `RS232` and see Table B.2 for commands of shutter controller.

- Baud rate 9600
- 8 data bits
- 1 stop bit
- No parity
- No flow control
- 8 commands are available
- 1 global address location for commands
- 8 local address location for commands

Channel#	Event	Decimal	Hex	Octal	Binary	ASCII
1	Open	64	40	100	01000000	@
1	Close	65	41	101	01000001	A
2	Open	66	42	102	01000010	B
2	Close	67	43	103	01000011	C
3	Open	68	44	104	01000100	D
3	Close	69	45	105	01000101	E
All	Open	70	46	106	01000110	F
All	Close	80	47	107	01000111	G

TABLE B.2: Shutter commands

FIGURE B.1: Overview of *LabVIEW* program for automated fs processing.

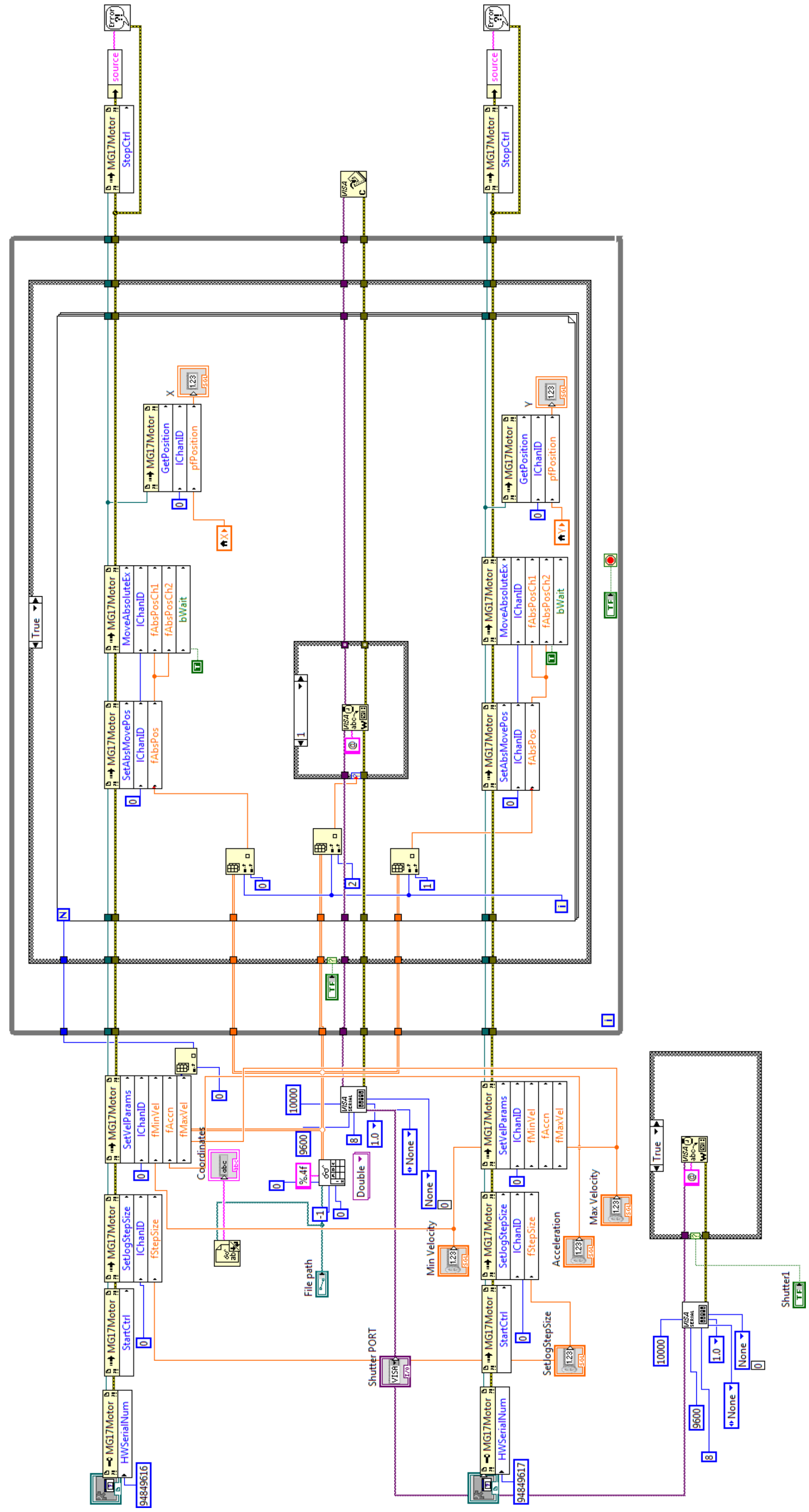


FIGURE B.2: Block diagram of the *LabVIEW* program.

Appendix C

Optical delay line software

C.1 Peizo electric stack actuator

Peizo electric stack actuators with strain gauge is used in the delay line to introduce time delay. Table C.1 shows the specifications of the PZT used.

Parameter	Value
Motorized axes	X, Y, Z
Travel range (open loop)	100 μm
Axial load capacity	40/40/32 N
Axial stiffness	1 $N/\mu m$
Vertical load capacity	30 N
Capacitance	1.8 μF
Close loop repeatability	30 nm
Closed loop resolution	4 nm
Closed loop travel	80 μm
Open loop resolution	0.4 nm
Resonant frequency at 105 g load	190/180/250 Hz
Resonant frequency at 300 g load	110/110/150 Hz
Resonant frequency at 80 g load	210/200/300 Hz
Resonant frequency unloaded	500/550/480 Hz
Weight	160 g

Table C.1: NPXYZ100SG specifications

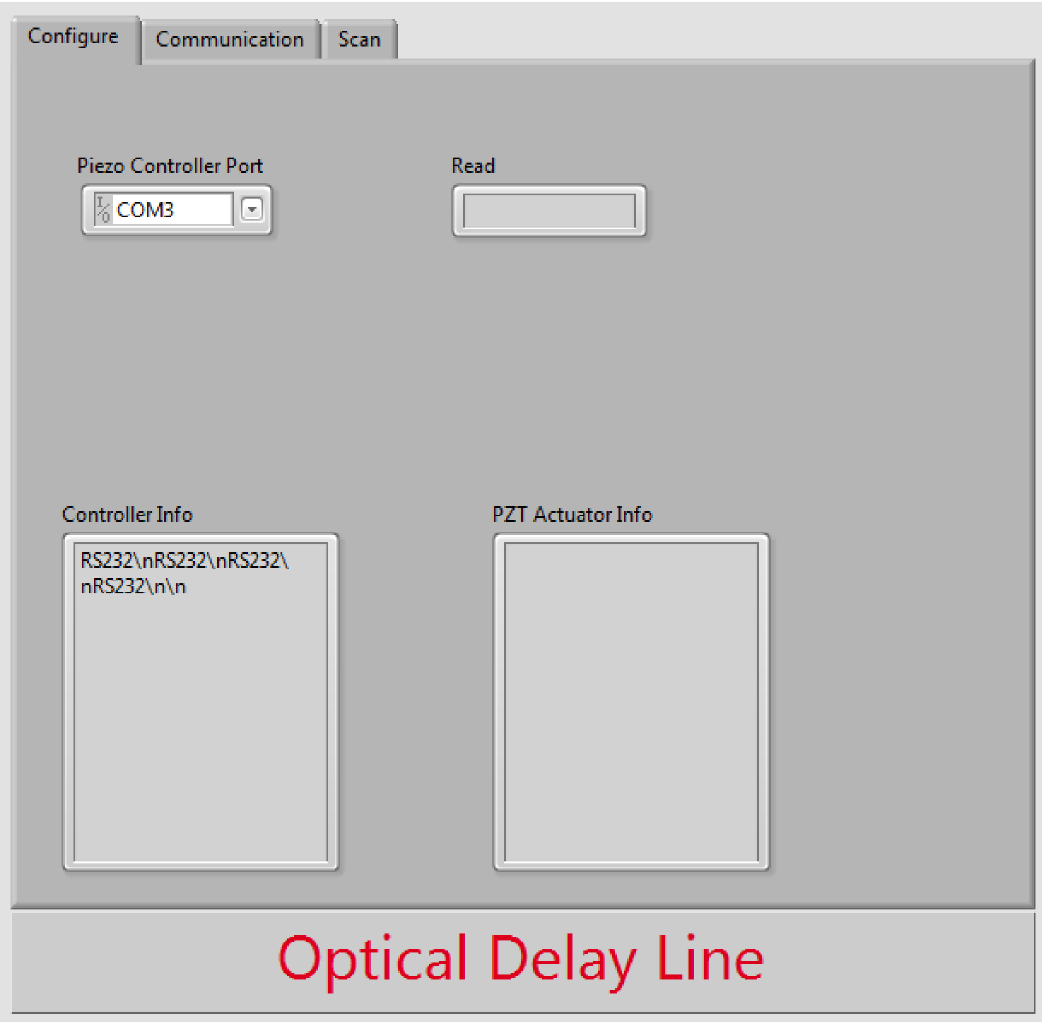


Figure C.1: Configure tab of the program

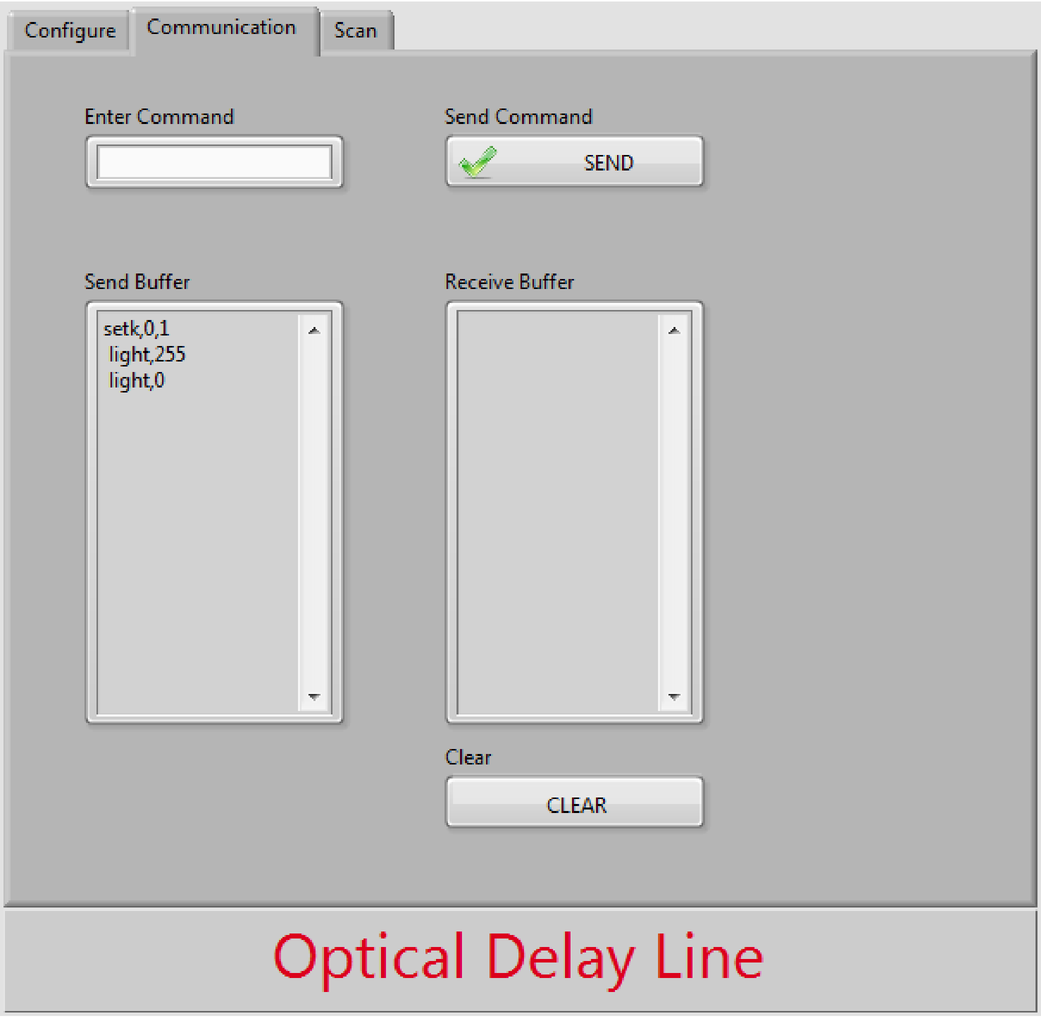


FIGURE C.2: Communication tab of the program

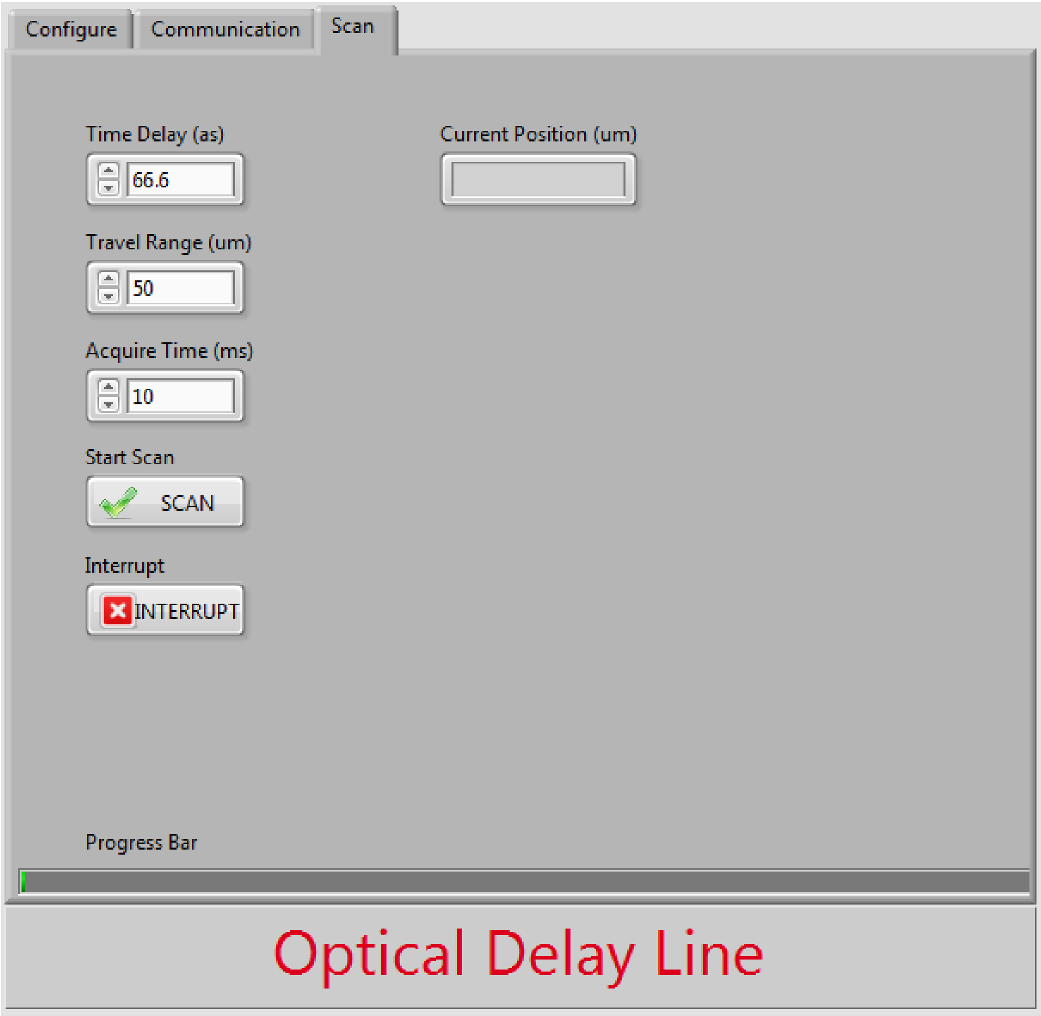


FIGURE C.3: The scan tab of the *LabVIEW* program of optical delay line. `Delay Time (as)` is optical delay step required in *as*. `Travel Range` is the scanning range in μm. `Acquire Time (ms)` is the time for which PZT stays at a position during scan. `SCAN` button initiates the optical delay scan with steps of `Delay Time`. `INTERRUPT` button to interrupt/stop scan the scan. `Progress Bar` shows the scanning progress.

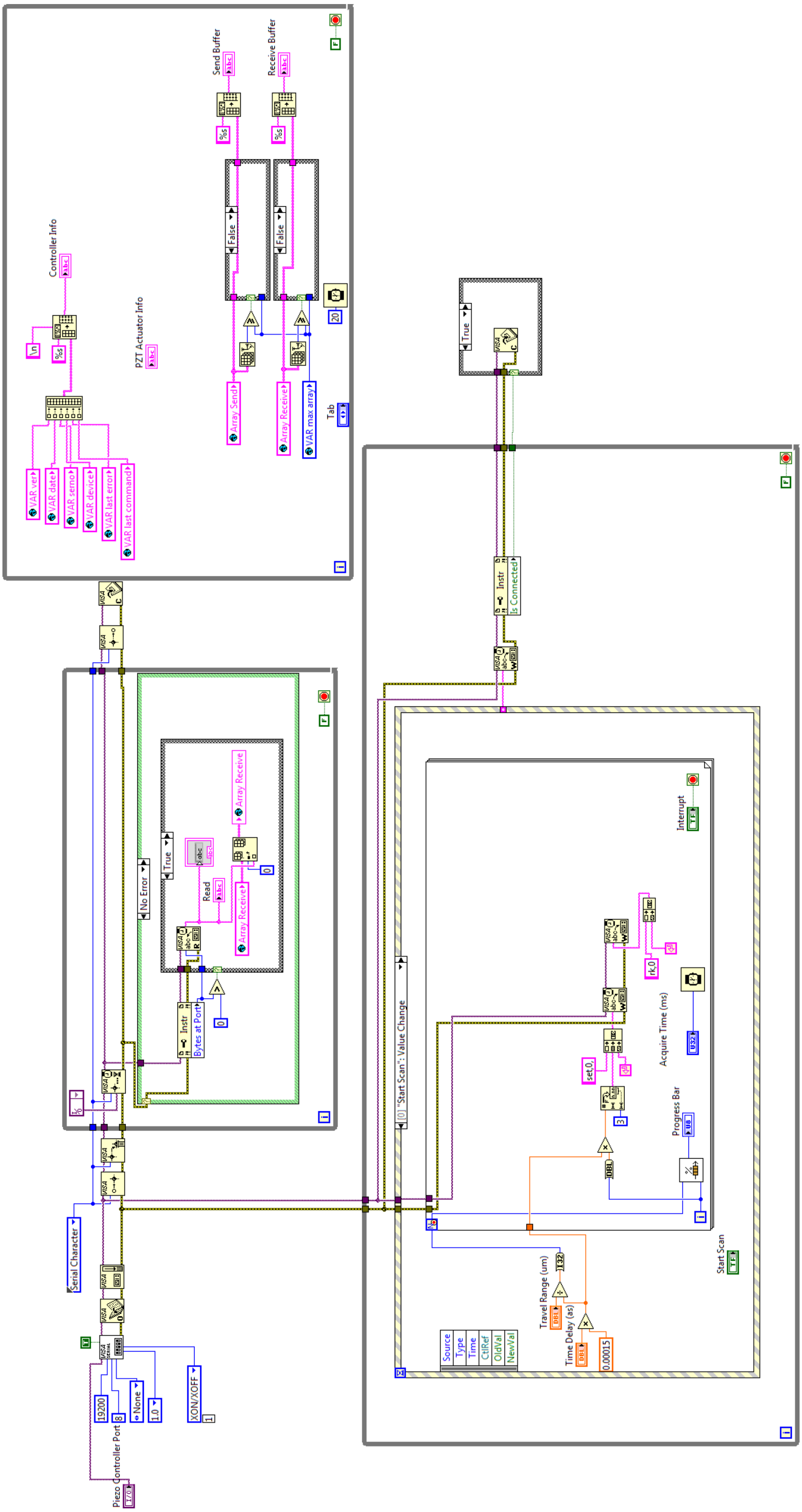


FIGURE C.4: Block diagram of the ODL *LabVIEW* program.

Bibliography

[1] R. Paschotta. *Optical delay lines*. Encylopedia of Laser Physics and Technology, 1 edition, October 2008. ISBN 978-3-527-40828-3. URL https://www.rp-photonics.com/optical_delay/_lines.html.

[2] Rick Trebino, Carl C. Hayden, Anthony M. Johnson, Wayne M. Simpson, and Alfred M. Levine. Chirp and self-phase modulation in induced-grating autocorrelation measurements of ultrashort pulses. *Optics Letters*, 15(19):1079–1081, 1990. URL https://www.osapublishing.org/ol/viewmedia.cfm?uri=ol-15-19-1079.

[3] Xiumei Liu, Mechael J. Cobb, and Xingde Li. Rapid scanning all-reflective optical delay line for real-time optical coherence tomography. *Optics Letters*, 29:80–82, 2004. URL https://www.osapublishing.org/ol/abstract.cfm?uri=ol-29-1-80.

[4] A. Zair, E. Mevel, E. Cormier, and E. Constant. Ultrastable collinear delay control setup for attosecond ir-xuv pump-probe experiment. *Optical Physics*, 35(5):A110–A115, May 2018. URL https://www.osapublishing.org/josab/abstract.cfm?uri=josab-35-5-A110.

[5] Andreas Tunnermann, Stefan Nolte, and Jens Limpert. Femtosecond vs picosecond laser material processing. *LTJ*, 7(1):34–38, January 2010. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/latj.201090006.

[6] John E. Grayson. *Python and Tkinter programming*. Manning, 2000. ISBN 9781884777813. URL https://www.manning.com/books/python-and-tkinter-programming.

[7] Martin C. Brown. *Python: The complete reference.* McGraw-Hill, 2018. ISBN 9789387572942. URL https://www.mheducation.co.in/9789387572942-india-python-the-complete-reference.

[8] Kent D. Lee. *Python programming fundamentals.* Springer, 2011. ISBN 978-1-84996-536-1. URL https://www.springer.com/in/book/9781849965378.

[9] John Essick. *Hands-on introduction to LabVIEW for scientists and engineers.* Oxford University Press, USA, 2018. ISBN 9780190853068. URL https://global.oup.com/academic/product/hands-on-introduction-to-labview-for-scientists-and-engineers-9780190853068?cc=in&lang=en&.