ME 395 AND CS 395 JOINT INTERNSHIP PROJECT REPORT

FABRICATION OF STRAIN SENSOR VIA EMBEDDED 3D PRINTING TECHNIQUE

Okan Arif Güvenkaya

26780

05.09.2022 - 16.09.2022

Sabancı University, Soft Robotics and Control Lab

Melih Türkseven

25.09.2022

Sabancı University

Faculty of Engineering and Natural Sciences

Major: Mechatronics Engineering

Double Major: Computer Science and Engineering

Minor: Physics

ABSTRACT

Sabancı University was established in 1999 in Turkey by Sabancı Foundation. The internship was done in Soft Robotics and Control Lab under the Faculty of Engineering and Natural Science of Sabancı University. There are several subfields of robotics. One of the latest famous ones is soft robotics. Soft robotics is an area that is interested in robot fabrication via compliant materials, not rigid ones. Recently, curiosity about soft robotics has created a new field of electronics which is called stretchable electronics. One example of these electronic devices is a strain sensor. In general, soft sensors are composed of two materials. One of them is deformable conducting material, the other one is inactive stretchable material. The conductive one is encapsulated by the second one. There are several production methods for creating these sensors. One of the newest techniques is embedded 3D printing. In this e-3D printing technique, the deposition nozzle extrudes the viscoelastic ink into the elastomeric reservoir. The reservoir material serves as and matrix element whereas the ink material is used as a resistive sensing element. The sensor will be ready when the reservoir material is cured. As a result, when the soft material stretch, the ink material will also stretch. Then due to strain, the resistance of the ink material will be changed. This is called a soft strain sensor. The project included the control and calibration of the 3D printing. Some background about g-code, python will be good for the understandability of the report.

TABLE OF CONTENTS

1. INTRODUCTION

Soft sensors are typically composed of a deformable conducting material patterned onto, attached to, or encapsulated within an inactive stretchable material. There are several production methods for creating these sensors. One of the newest techniques is embedded 3D printing. Viscoelastic ink is extruded through a deposition nozzle directly into an elastomeric reservoir in this e-3D printing technique. The ink is a resistive sensing element, while the reservoir serves as a matrix element. The sensor will be ready when the reservoir material is cured. As a result, when the soft material stretch, the ink material will also stretch. Then due to strain, the resistance of the ink material will be changed. This is called a soft strain sensor. This report explains the control and calibration process of the 3D embedded printer and dispenser in the Soft Robotics and Control Lab under the Faculty of Engineering and Natural Science of Sabancı University. During the internship, the control and calibration of the 3D printer, connection, and control of the dispenser were done. This report includes details of them, the experience of the intern, and conclusions of the project.

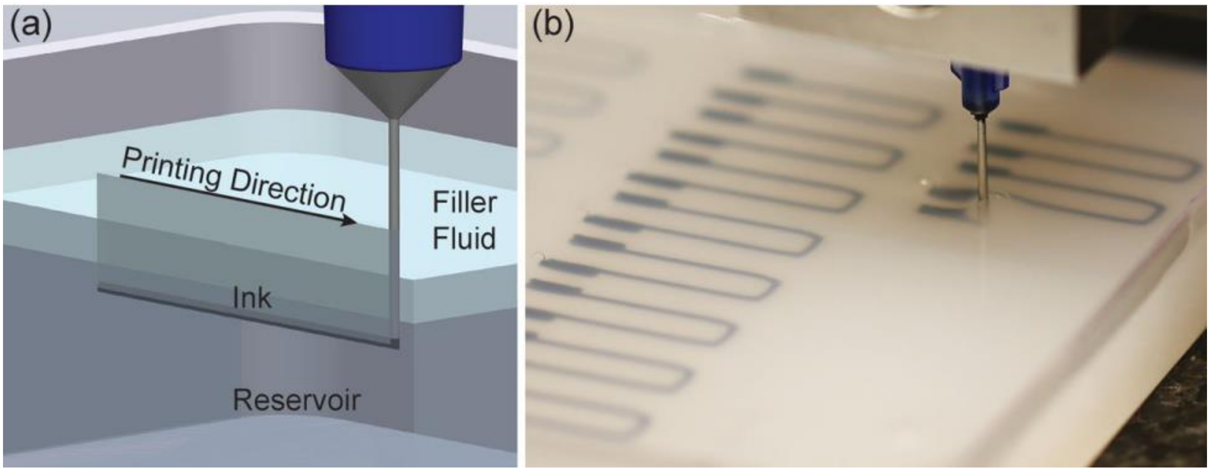


Figure 1: Embedded 3D Printing

2. COMPANY INFORMATION

Full Title of the Company: Sabancı University

Address of the Company: Sabancı University Tuzla Campus, Türkiye, İstanbul, Tuzla, Orta Mahalle, Üniversite Caddesi, No:27

Contact Telephone of the Company: +90(216)4839000

Web page of the Company: https://www.sabanciuniv.edu/en

At 1994, the Sabancı Group decided to open a world university. The Sabancı University was established in 1999 by the Sabancı Foundation. From 1999, the students started to enrolled the university.

Faculties of The University:

- Faculty of Engineering and Natural Sciences (FENS)
- Sabancı Business School (SBC)
- Faculty of Arts and Social Sciences (FASS)

Industry of the Company: Education

3. PROJECT BACKGROUND

3.1 Department Information

Soft robotics is an area that is interested in robot fabrication via compliant materials, not rigid ones. Recently, curiosity about soft robotics has created a new field of electronics which is called stretchable electronics. The Soft Robotics and Control Lab at Sabancı University is working on projects related to the Soft Robotics area. The lab work under one of the mechatronic academicians of Sabancı university Melih Türkseven. There are master's and doctorate students of him are working on the projects. Milad Hayati is one of them and with who I worked closely during the internship. Detailed information about them are following:

- **Name:** Melih Türkseven, Milad Hayati
- **Email:** melih.turkseven@sabanciuniv.edu, milad.hayati@sabanciuniv.edu
- **Title of the People:** Faculty Member, Master Student

3.2 Status of the Project and/or the Problem at the Beginning

The project aims to produce a strain sensor via an embedded 3D printing method. In the beginning, lab has a 3D printer, dispenser, elastomer material, and ink material. However, the 3D printer should need to be controlled by the user and need to be calibrated. After then the strain sensors will be produced.

3.3 Motivation and/or Problem Definition

As mentioned in the previous part, there was a 3D printer in the lab. The model of the 3D printer was Ender Pro 5. The printer is working via g-code, so at first, it was required to learn g-code. And printer worked with Raspberry PI. So it also needed to be learned. Also, the connection and usage details of the dispenser should be learned. At the end of the internship, the lab expects from intern to make a ready 3D printer that is controlled and calibrated. But during this process, toothpaste was used as an elastomeric material, since it is cheap and easier than carbon resin for the trial process.

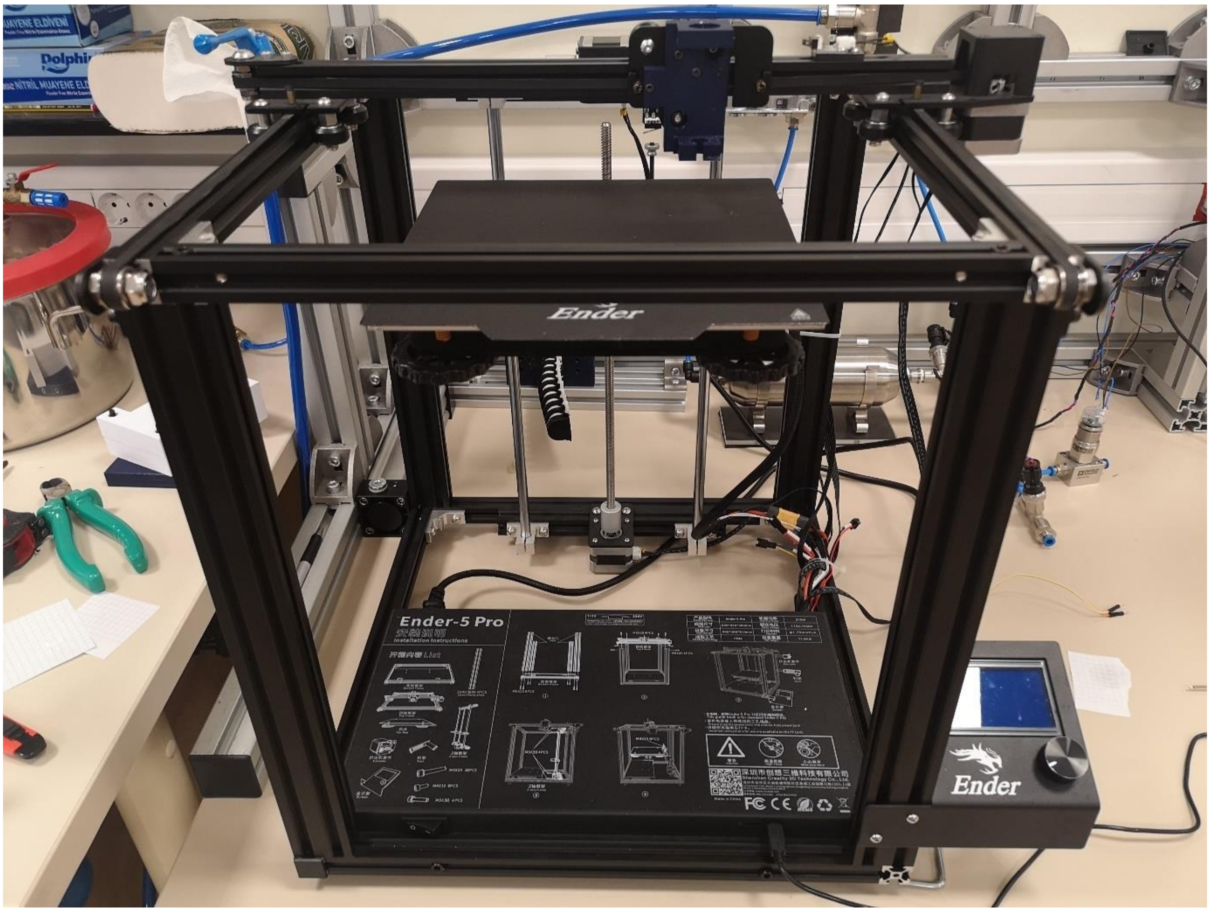


Figure 2: Ender 5 Pro 3D Printer

Figure 3: Fisnar DC100 Dispenser and Syringe Kit

3.4 Related Literature

The integration of rigid electronics into stretchable matrices is hard and not that easy to use. The soft sensors spring from there. In a strain sensor, the conductive material is encapsulated within stretchable materials. Some of the biggest usage fields of strain sensors are wearable electronics and human machine interfaces. Papers related to fabrication of strain sensor via embedded 3D printing techniques were read. And they were the major theoretical sources for strain sensors during the internship.

Figure 4: Strain Sensor usage that is fabricated by E-3D Printing:

Tutorials about g-code were watched in YouTube from different channels. One of the best of them was Makers Mashup: https://www.youtube.com/c/MakersMashup

Also, for dispenser settings about Fisnar DC 100, the brand itself has a YouTube channel about how to use it. https://www.youtube.com/channel/UC2Iwz6dabN5ITTZsgbDpUdQ

4. INTERNSHIP PROJECT

4.1 Project Objective

The main project objective is the fabrication of strain sensors by a method called embedded 3D printing. There was an Ender Pro 5 3D printer in the lab for embedded 3D printing. The aim is to make this printer ready for embedded 3D printing. The ink material is extruded via a syringe that is connected to Fisnar DC100 High Precision Dispenser. The printer is actuated by Raspberry Pi Model 4B.

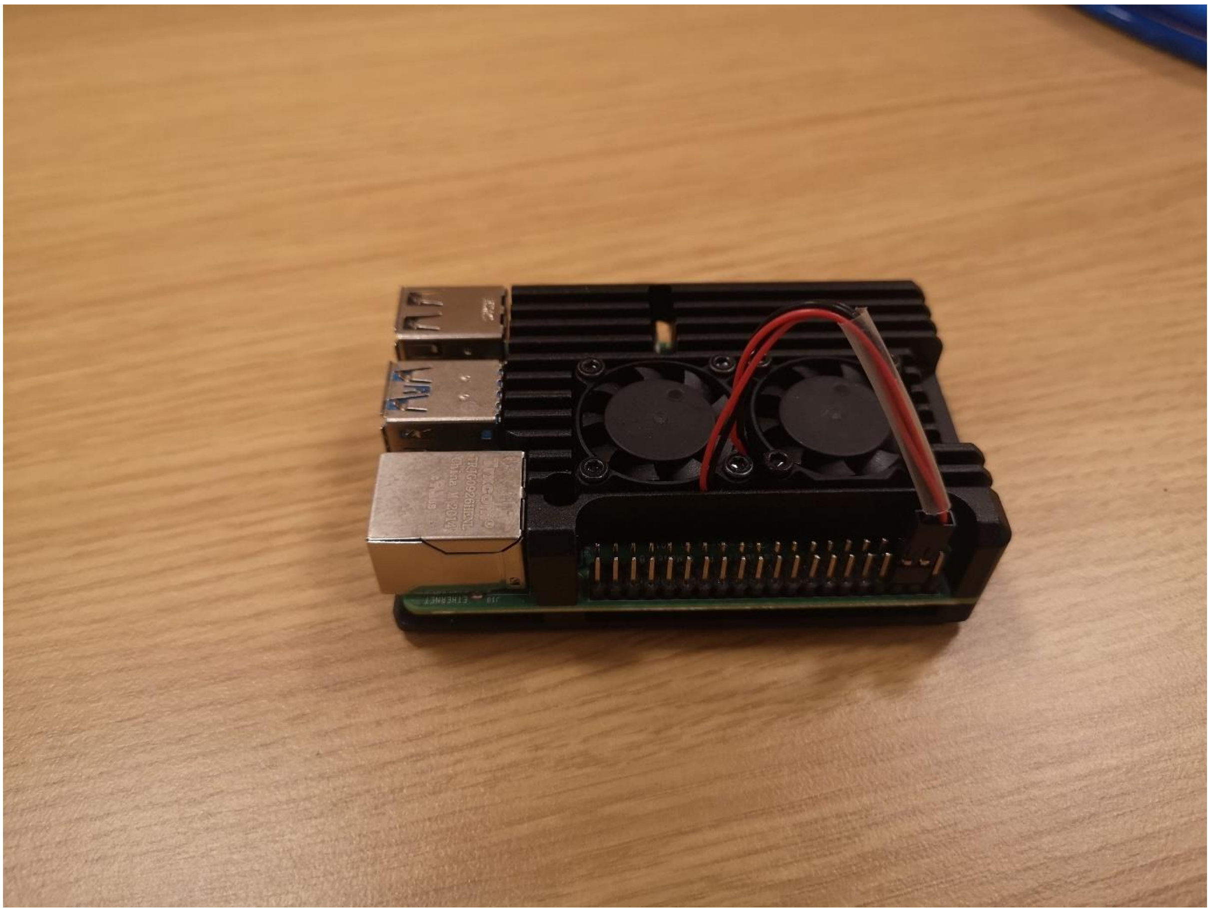


Figure 5: Raspberry Pi Model 4B

The printer will be controlled by code written in python and the help of the G-code method. The project aims to write python code via Raspberry Pi and control the action of the printer and dispenser. Also, there is a part that connects the syringe to the 3D printer, it is called a syringe holder. The design of the syringe holder also needs to be done, printed, and assembled for the printer. Additionally, after all, the works mentioned above, the calibrations need to be done for the printer. For 2 week duration, toothpaste was used for ink material since it is easier to use and cheaper than carbon resin. The usage of the carbon resin and fabrication of the strain sensor will be the next stage of the project. It was not included in the first step.

4.2 My Responsibilities

For the soft sensor production method embedded 3D printer, the ink materials are highly priced such as carbon. Therefore, in the first stage, the project aims to use toothpaste as an ink material because it has similar viscous qualities to carbon. In the project, the lab expects the intern to design a ready 3D printer which is working properly with a dispenser where toothpaste is used as ink material. In other words, the usage of the dispenser will be learned, holder geometry will be designed, python code will be written, and calibration of the 3D printer will be done at the end of the project.

4.3 Methodology / Tools

Tools and their models were as followings:

- The printer: Ender Pro 5.
- The dispenser: Fisnar DC100/DC200 High Precision Dispenser
- Computer: Raspberry Pi Model 4B
- Syringe Barrels, End Caps, Pistons, Syringe Barrel adapters and Accessories, Manual Syringe Plunger: Fisnar
- Ink Material: toothpaste

Software's:

To control the printer, the intern was required to learn g-code. For the making easy of usage of the printer, python was used and written in google collab. During the internship, raspberry pi gave a buffer error. The intern found a new method to use and control the printer. During this process, the intern learned some of the programs named Cura and Gcode-Sender. During the design process, the part that assembles the syringe to the printer was designed in Solidworks.

4.4 Expected Outcome and Deliverables

Lab expects from intern to prepare a 3D printer which is ready for an embedded 3D printer for fabrication of strain sensor. The limitations are required defined in software so the user cannot go further into the printer's bad area. The only thing that the user can do is use the simplified python functions for printing what he or she wants.

4.5 Details

At first, the concept of strain sensors and embedded 3D printing should be understood. For this reason, literature reviews were done on the internet and papers related to fabrication of strain sensor via embedded 3D printing techniques were read. And they were the major theoretical sources for strain sensors during the internship Additionally, to figure out how to control 3D printers, tutorials about g-code and Raspberry Pi were watched on the internet.

A dispenser is a tool that pushes the ink material from a syringe and causes the flow of the ink material to the outside of the syringe. This process is printing part of the printer. The dispenser was controlled by a relay, so the working principle of the relay also was learned.

After the literature review, intern got familiar with g-code, raspberry pi, and the 3D printer interface. To control the printer easier, some of the python functions are written by using some of the basic g-code comments that were required for the internship.

The g-code comments that were used in python functions:

- G28: Home Operation
- G0: Absolute motion

- G90: Absolute Positioning
- G91: Relative Positioning
- M400: Waiting previous point to be completed

Python Functions for linear motion and relay on off for printing:

- G0_ABS: The printhead moves the desired location directly with respect to the origin.
- G0_REL: The printhead moves the desired location directly with respect to the last position.
- G0_ABS: The printhead moves the desired location directly with respect to the origin.
- M42: Control the relay switching. This was for printing on off.

After the basics of linear motion control of the printer, some of the circular motion functions were also coded.

- G2_CIRCLE: The printhead follows a circle in a clockwise direction.
- G3_CIRCLE: The printhead follows a circle in a counter clockwise direction.
- G2_Arc: The printhead follows an arc path in a clockwise direction.
- G3_Arc: The printhead follows an arc path in a counterclockwise direction.

Then, the laying the foundations of the motion and printing code, the time comes for the usage of the dispenser part. In the beginning, the problem was how to fill the syringe with ink material. You cannot directly fill it from behind because you need to not leave any space in the syringe while filling the operation. This space causes non-desired printing errors. The techniques were found out and the syringe was filled with toothpaste.

After the filling process, it was time for understanding the working principle and get familiar with the dispenser. The datasheet of the dispenser was read and all the functions of the Fisnar DC100 were learned. There are different working modes of a dispenser, the purge mode was the best option for the project purpose.

There are 2 kinds of tips in the lab for the syringe. These are Blunt End Dispensing Tips (BET) and Tapered Dispensing Tips (TT). Tapered Dispensing Tips is what we will use because it is designed for high viscous materials.

Figure 6: Blunt End Dispensing Tip

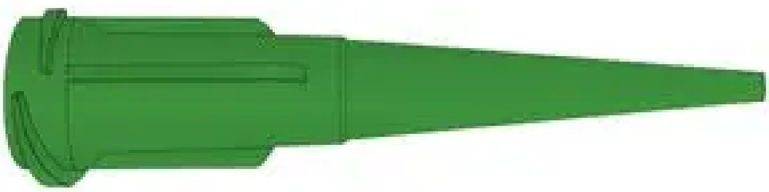


Figure 7: Tapered Dispensing Tip

After that, the required dimensions were measured and the proper 3D cad design is done in Solidworks for a syringe. It was 3D printed in the 3D printer in the lab.

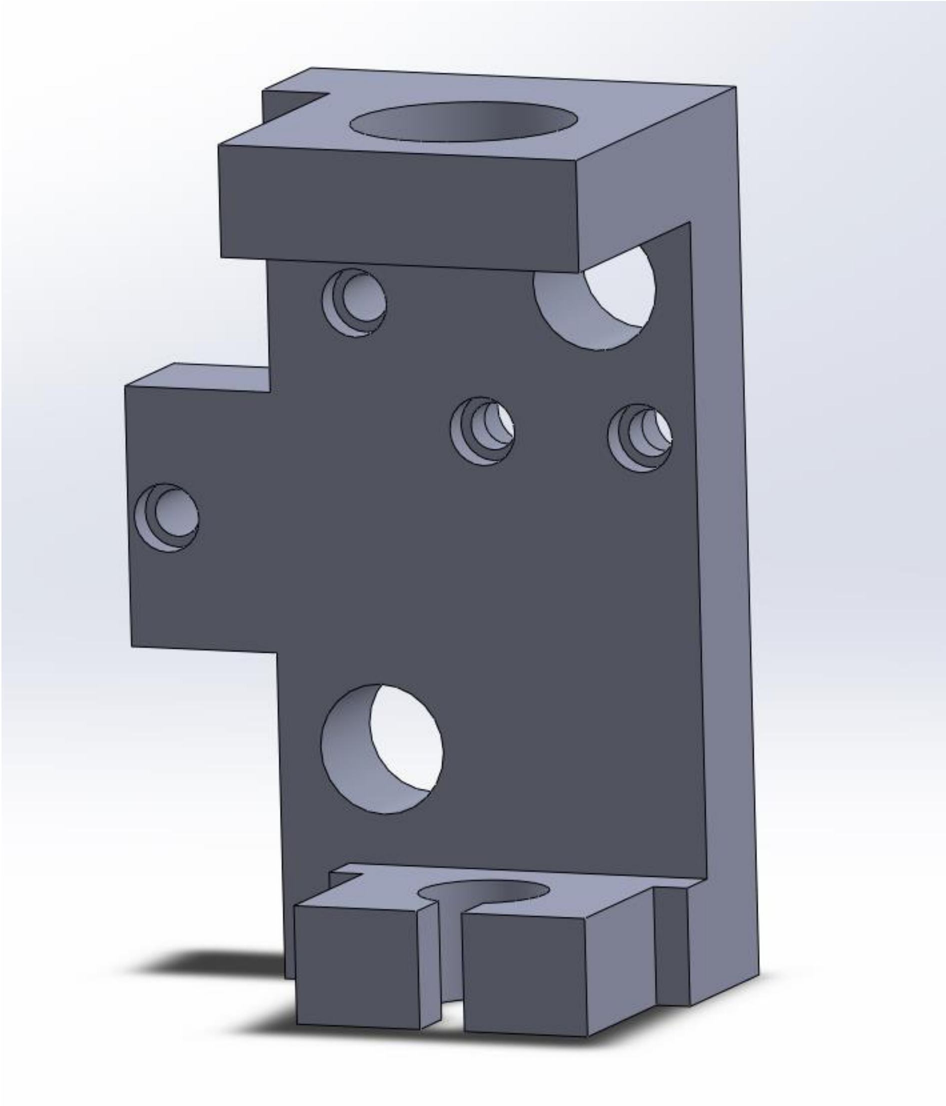


Figure 8: Syringe Holder 3D Design

Then, the code was updated based on the boundaries of the printer bed. Due to the design of the ender pro itself and holder design, sometimes the printhead can move out of the printer bed boundaries. Remapping and boundary check functions have been added to the code. As a result, if the user wants to move out of the borders, the code will give an error and the process will not start.

After updates on the code, the connections of the dispenser were done. And there were 4 important calibration parameters. These are:

1- Tip Bed Distance (TBD): Z-axis distance between the end of the syringe tip and printer bed.

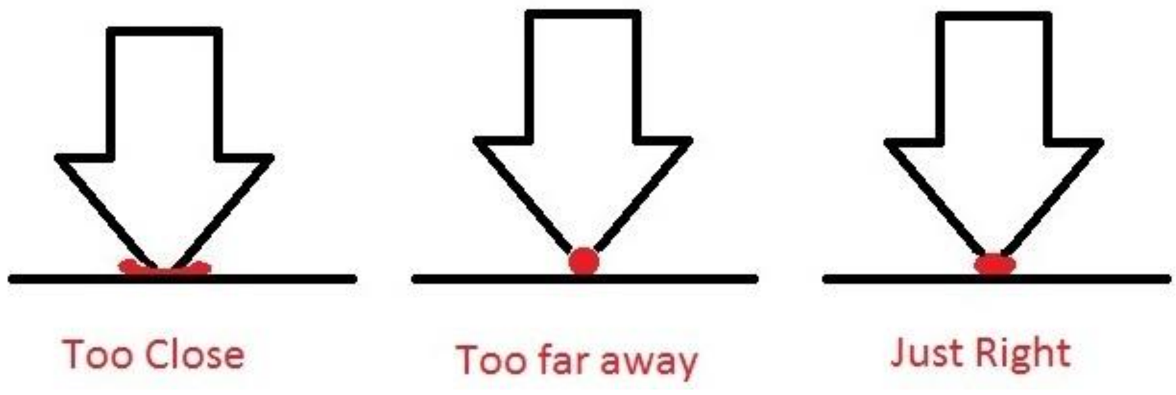


Figure 9: Tip Bed Distance Configuration Samples and Results

2- Feed Rate: The speed of the printhead motion.

3- Pressure: The output pressure which is given to the syringe on the printer.

4- Vacuum Pressure: Vacuum pressure affects retract when the printing is finished. Prevent the flow of ink material when the printing is stopped.

Lots of trials were done about the above parameters and an excel table was prepared. As a result, ideal values were found as followings:

- TBD: 0.2 mm
- Feed Rate: 500 mm/min for the printing process, 1500 mm/min for non-printing move
- Pressure: 15 psi for toothpaste
- Vacuum: 1.6 psi for toothpaste. (The observation of vacuum was not good due to the physical qualities of the toothpaste, so it was concluded that another material will be better for trials. In the end, it is mentioned. The Vaseline will be tried in the lab for the calibration which is more similar to viscous quantities as carbon resin)

After a while, the delay problem happened. When the motion of the printhead, the delay happens. This was caused due to buffer problem. New methods and software were tried to find out instead of using raspberry Pi. The Cura was one of them. It was learned but it does not let the user enter a group of g codes into it. Finally, a software called g-code-sender that is produced by google chrome was found and learned. It allows to user to use the g-code file. So all the code written in the raspberry Pi was updated an rewritten in google collab. And more functions were added. In the end, the user can write the motion code but just use simplified python functions that were written by the intern. The user does not need to know any g code. If there is any limitation problem, the code gives an error. If there is no limit problem, the code printout each motion to the user. Additionally, the code plots a graph that shows the expected printhead motion and will print the shape on the screen. And also it produces a g-code file and downloads it automatically. When the user uploads the file to the g-code sender, it works properly.

```
Relative Linear Motion is working with distances: X = 211.0 Y = 171.0 Z = 0.20000000000000018
Current Location with respect to redefined origin:
X:  211.0
Y:  171.0
Z:  0.20000000000000018
#############################################
Relative Linear Motion is working with distances: X = 208.0 Y = 170.0 Z = 0.20000000000000018
Current Location with respect to redefined origin:
X:  208.0
Y:  170.0
Z:  0.20000000000000018
#############################################
Printing: OFF
#############################################
Relative Linear Motion is working with distances: X = 214.0 Y = 170.0 Z = 2.2
Current Location with respect to redefined origin:
X:  214.0
Y:  170.0
Z:  2.2
#############################################
Relative Linear Motion is working with distances: X = 214.0 Y = 170.0 Z = 0.20000000000000018
Current Location with respect to redefined origin:
X:  214.0
Y:  170.0
Z:  0.20000000000000018
#############################################
Expected path of the printhead and will be printed shape is working
```

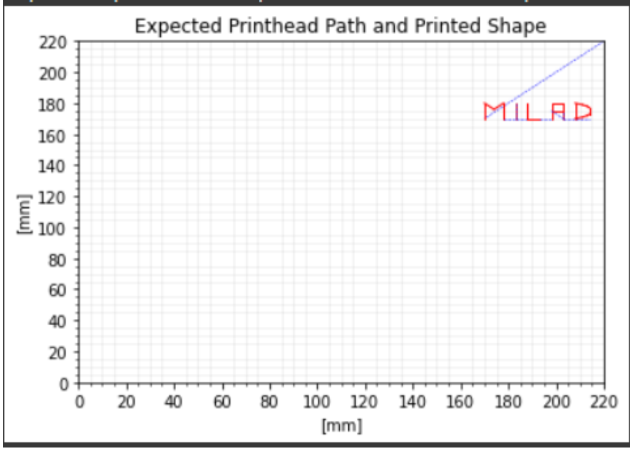


Figure 10: Some part of the output of the successful printing commands

```
ValueError                                Traceback (most recent call last)
<ipython-input-1-93854e146cef> in <module>
    560 # ACTION CODE
    561 ##################################################################################################
--> 562 LinearMotionABS(speedValueMove, 250, 170, 10)
    563 LinearMotionABS(speedValueMove, 170, 170, TBD)
    564 # Write MILAD

2 frames
<ipython-input-1-93854e146cef> in OutOfBorderErrorDisplay(axis)
    173     errorSuggestionMessage_2 = "Caution: There cannot be any point out of cube that edges on " + MaxMinLimitsStr
    174     wholeErrorMessage = errorMessage + "\n" + errorMessageDescription + "\n" + errorSuggestionMessage + "\n" + errorSuggestionMessage_2
--> 175     raise ValueError(wholeErrorMessage)
    176
    177 # Check the target location wheter it is inside or outside the boundary of the printer bed

ValueError: ERROR: X AXIS VALUE IS OUT OF LIMITATION
Error Description: Your drawing borders are out of printer bed borders in x direction.
Suggestion: Control the current location and your drawing point.
Caution: There cannot be any point out of cube that edges on {(0,0,0),(0,0,220),(0,220,0),(0,220,220),(220,220,0),(220,220,220),(220,0,0),(220,0,220)}
```

Figure 11: Sample of out of boundary error

4.6 Results

The project itself is not a short-term project, it was a long-term project. The intern was responsible for the control and calibration of the 3D printer. The control and calibration processes were done successfully by the intern during the 10 working days internship period. Any user can use easily the 3D printer that is working with the dispenser.

5. INTERNSHIP EXPERIENCE

5.1 Details

It was my first experience working in a lab and being part of academic research. Lab working and working in academic research was good. I learned how to be part of any project in academy format. My first 30 work days internship discouraged me from working under 08:00-17:00 companies. However, this lab experience motivated me to work in labs and be a part of the academy.

5.2 Relation to Undergraduate Education

I used my Solidworks skills while designing a holder for a syringe and I learned Solidworks in the ENS209 course and Sabancı Robotic Club. I used my python background from IF100 and CS412. I also use the coding algorithm techniques that I learned from CS204 and CS300.

Some special courses about CNC machine control or soft robotics would be awesome.

5.3 Difficulties

The hose diameter for the syringe connection was nearly 4mm in the Fisnar syringe kit. However, the output port of the dispenser was 6mm in diameter. At first, when we plug the hose into the dispenser and give the pressure, the leakage happened. Also, some unexpected decreases happened when the pressure was given. We tried to understand the problem. When we realize the situation we use a converter that is 4mm to 6mm and adds a small 6mm hose. Then the problem is solved.

5.3 A Typical Day

In general, I came to the lab at 10:30. Before starting to work on anything, I do a to-do list for that day. Then I started to work. I worked until 18:40 each day. Also before leaving, I took detailed notes about what I have done today or problems, etc. I made my launch in between 10:30-18:40, the exact time depended on that day's work.

6. CONCLUSIONS

The internship project was related to the preparing a 3D printer for fabrication of strain sensors via method called embedded 3D printing. I have learned how to use and control CNC and 3D printer type machines by g-code. Additionally, I experienced first time work with dispenser and a pressure system. I used my previous CS course knowledge and code the printer for easy usage of the user.

7. RECOMMENDATIONS

Especially in research areas, theoretical knowledge became more important. If you have a better background, then the internship becomes more effective and enjoyable. Therefore, I suggest furthering students listen to their university courses carefully. When you do that, the process and what you are doing makes more sense in your mind. You will be more productive in the internship project. The second recommendation is that do not hesitate to ask questions. The aim of the internship is learning. And the people who you work with are responsible to teach you something. To make the internship more educational, ask questions about everything that confuses your mind. The third and last suggestion is to take notes per day. These notes will make your internship more effective and will help you while you are writing your internship report.

8. REFERENCES

Muth, J. T., Vogt, D. M., Truby, R. L., Mengüç, Y., Kolesky, D. B., Wood, R. J., & Lewis, J. A. (2014). Embedded 3D printing of strain sensors within highly stretchable elastomers. *Advanced materials*, *26*(36), 6307-6312.

9. APPENDICES

```python
# IMPORTING LIBRARIES
import numpy as np    # To use array and array functions
from google.colab import files
import matplotlib.pyplot as plt

# GLOBAL VARIABLES
# All G-code String
G_CODE = ""

# Current Printing Mode, off:0, on:1
PRINTING_MODE = 0

# Current Printhead Locations with respect to the absolute origin
X_CURRENT = 0    # current location of printhead in x axis
Y_CURRENT = 0    # current location of printhead in y axis
Z_CURRENT = 0    # current location of printhead in z axis

# Absolute Minimum and Maximum Locations for Printer
X_ABS_MIN = 0;    # min location of printer in x axis
X_ABS_MAX = 220;    # max location of printer in x axis
Y_ABS_MIN = 0;    # min location of printer in x axis
Y_ABS_MAX = 220;    # max location of printer in x axis
Z_ABS_MIN = 0;    # min location of printer in x axis
Z_ABS_MAX = 220;    # max location of printer in x axis

# Relative Minimum and Maximum Locations for Printer
X_MIN = 0;    # min location of printhead in x axis
X_MAX = 220;    # max location of printhead in x axis
Y_MIN = 0;    # min location of printhead in y axis
Y_MAX = 220;    # max location of printhead in y axis
Z_MIN = 0;    # min location of printhead in z axis
Z_MAX = 220;    # max location of printhead in z axis

# To Show the expected Printhead motion and Printed Shape
ALL_PRINTHEAD_MOTION_COORDINATIONS = []    #last -1: printng off, -2: printing on

"""
Due to 3D design of the syringe holder, some gaps occur. As a result, it is not possible to use whole absolute printer area
if your printhead point an outside of the bed at home position (x=220, y=220) or point at x=0, y=0.

X AND Y CALLIBRATION
Assume you 3D printed your syringe holder. Use G28_Home() function and home your printer. Measure the gap between
tip and end of the printer bed. If tip is outside the bed use positive sign, otherwise use negative sign.

Make same work with using G0_ABS(0,0,0) function.

Z CALLIBRATION
You can configurate your Z axis by phisically. Turning screw on the printer.
"""
# Gaps Between Tip and Printer Bed Boundaries
# Gaps at Origin (0,0,0)
x_origin_gap = 0;    # gap between tip and printer bed boundary at x axis
y_origin_gap = 0;    # gap between tip and printer bed boundary at y axis
z_origin_gap = 0;    # gap between tip and printer bed boundary at z axis

# Gaps at Upper Limit (G28_Home)
x_upper_gap = 0;    # gap between tip and printer bed boundary at x axis
y_upper_gap = 0;    # gap between tip and printer bed boundary at y axis
z_upper_gap = 0;    # gap between tip and printer bed boundary at z axis

"""
X AND Y CALLIBRATION
If the gap is negative there will be no problem. The problem occurs where these gaps are positive.
There are 2 cases for x and y axis:
Case 1 X: x_origin_gap > 0: The relative max will be reduced, absolute min will be the gap.
Case 2 X: x_upper_gap > 0: The relative max will be reduced, absolute max will be reduced.
Case 1 Y: y_origin_gap > 0: The relative max will be reduced, absolute min will be the gap.
Case 2 Y: y_upper_gap > 0: The relative max will be reduced, absolute max will be reduced.
"""
```

```python
if x_origin_gap > 0:
    X_MAX = X_MAX - x_origin_gap
    X_ABS_MIN = x_origin_gap
if x_upper_gap > 0:
    X_MAX = X_MAX - x_upper_gap
    X_ABS_MAX = X_ABS_MAX - x_upper_gap

if y_origin_gap > 0:
    Y_MAX = Y_MAX - y_origin_gap
    Y_ABS_MIN = y_origin_gap
if y_upper_gap > 0:
    Y_MAX = Y_MAX - y_upper_gap
    Y_ABS_MAX = Y_ABS_MAX - y_upper_gap

# Form string of limits for later informing functions to user
leftDownLowLimitStr = "(" + str(X_MIN) + "," + str(Y_MIN) + "," + str(Z_MIN) + ")"
leftDownHighLimitStr = "(" + str(X_MIN) + "," + str(Y_MIN) + "," + str(Z_MAX) + ")"
leftUpLowLimitStr = "(" + str(X_MIN) + "," + str(Y_MAX) + "," + str(Z_MIN) + ")"
leftUpHighLimitStr = "(" + str(X_MIN) + "," + str(Y_MAX) + "," + str(Z_MAX) + ")"
rightUpLowLimitStr = "(" + str(X_MAX) + "," + str(Y_MAX) + "," + str(Z_MIN) + ")"
rightUpHighLimitStr = "(" + str(X_MAX) + "," + str(Y_MAX) + "," + str(Z_MAX) + ")"
rightDownLowLimitStr = "(" + str(X_MAX) + "," + str(Y_MIN) + "," + str(Z_MIN) + ")"
rightDownHighLimitStr = "(" + str(X_MAX) + "," + str(Y_MIN) + "," + str(Z_MAX) + ")"
MaxMinLimitsStr = "{"+leftDownLowLimitStr+","+leftDownHighLimitStr+","+leftUpLowLimitStr+","+leftUpHighLimitStr+","+rightUpLowLimitStr+","+rightUpHighLimitStr+","+rightDownLowLimitStr+","+rightDownHighLimitStr+"}"
```

```python
# Absolute move to X Y Z coordinates with speedValue
# X: Target coordinate in x axis in mm
# Y: Target coordinate in y axis in mm
# Z: Target coordinate in z axis in mm
def G0_ABS(speedValue = 500, X = 0, Y = 0, Z = 0):
    global X_CURRENT, Y_CURRENT, Z_CURRENT, G_CODE
    # converting inputs to string format
    X_str = str(X)
    Y_str = str(Y)
    Z_str = str(Z)
    speedValue_str = str(speedValue)

    # Update current location of the printhead
    X_CURRENT = X    # current location of printhead in x axis
    Y_CURRENT = Y    # current location of printhead in y axis
    Z_CURRENT = Z    # current location of printhead in z axis

    # Form string format of g-code
    G_CODE = G_CODE + "\n" + "G90"    # Absolute positioning
    G_CODE = G_CODE + "\n" + "G0 " + "F" + speedValue_str + " X" + X_str + " Y" + Y_str + " Z" + Z_str    # Move to entered location
    G_CODE = G_CODE + "\n" + "M400"    # Wait until the previous g-code is completed
```

```python
# Relative move to X Y Z coordinates with respect to the last position with speedValue
# X: Replacement in x axis in mm with respect to the last position
# Y: Replacement in y axis in mm with respect to the last position
# Z: Replacement in z axis in mm with respect to the last position
def G0_REL(speedValue = 500, X = 0, Y = 0, Z = 0):
    global X_CURRENT, Y_CURRENT, Z_CURRENT, G_CODE
    # converting inputs to string format
    X_str = str(X)
    Y_str = str(Y)
    Z_str = str(Z)
    speedValue_str = str(speedValue)

    # Update current location of the printhead
    X_CURRENT = X_CURRENT + X;    # current location of printhead in x axis
    Y_CURRENT = Y_CURRENT + Y;    # current location of printhead in y axis
    Z_CURRENT = Z_CURRENT + Z;    # current location of printhead in z axis

    # Form string format of g-code
    G_CODE = G_CODE + "\n" + "G91"    # Relative ositioning
    G_CODE = G_CODE + "\n" + "G0 " + "F" + speedValue_str + " X" + X_str + " Y" + Y_str + " Z" + Z_str    # Move to entered location
    G_CODE = G_CODE + "\n" + "M400"    # Wait until the previous g-code is completed
```

```python
# Setting state of Pin P to S state with pin mode T
# P: Pin number
# S: State, for PWM pins 0<S<255
# T: Pin mode: T0=input, T1=output, T2=input_pullup, T3=input_pulldown
def M42(P, S, T):
    global G_CODE
    # converting inputs to string format
    P_str = str(P)
    S_str = str(S)
    T_str = str(T)

    # Form string format of g-code
    G_CODE = G_CODE + "\n" + "M42" + " P" + P_str + " S" + S_str + " T" + T_str

# Home printhead
def G28_Home():
    global X_CURRENT, Y_CURRENT, Z_CURRENT, X_ABS_MAX, Y_ABS_MAX, Z_ABS_MAX, G_CODE
    # Update current location of the printhead
    X_CURRENT = X_ABS_MAX
    Y_CURRENT = Y_ABS_MAX
    Z_CURRENT = Z_ABS_MIN

    # Form string format of g-code
    G_CODE = G_CODE + "\n" + "G28"    # G28: Home Printhead
    G_CODE = G_CODE + "\n" + "M400"    # Wait until the previous g-code is completed

# Print error and suggestion message about out of boundary.
# axis: axis that has an error, it should be in lower case
def OutOfBorderErrorDisplay(axis):
    axisCapital = axis.upper()
    errorMessage = "ERROR: "+ axisCapital + " AXIS VALUE IS OUT OF LIMITATION"
    errorMessageDescription = "Error Description: Your drawing borders are out of printer bed borders in " + axis + " direction."
    errorSuggestionMessage = "Suggestion: Control the current location and your drawing point."
    errorSuggestionMessage_2 = "Caution: There cannot be any point out of cube that edges on " + MaxMinLimitsStr
    wholeErrorMessage = errorMessage + "\n" + errorMessageDescription + "\n" + errorSuggestionMessage + "\n" + errorSuggestionMessage_2
    raise ValueError(wholeErrorMessage)
```

```python
# Check the target location wheter it is inside or outside the boundary of the printer bed
# Location is in the boundary, returns true
# Location is out of the boundary, returns false and print error message
def BoundryCheck(X = 0, Y = 0, Z = 0):
    global X_CURRENT, Y_CURRENT, Z_CURRENT, X_ABS_MAX, Y_ABS_MAX, Z_ABS_MAX, X_ABS_MIN, Y_ABS_MIN, Z_ABS_MIN
    boundaryCheck = True
    if X < X_ABS_MIN or X > X_ABS_MAX:
        OutOfBorderErrorDisplay("x")
        boundaryCheck = False
    elif Y < Y_ABS_MIN or Y > Y_ABS_MAX:
        OutOfBorderErrorDisplay("y")
        boundaryCheck = False
    elif Z < Z_ABS_MIN or Z > Z_ABS_MAX:
        OutOfBorderErrorDisplay("z")
        boundaryCheck = False
    return boundaryCheck
```

```python
# Plot the epected printhead path and will be printed shape
def DrawPrintheadMotionsHelper(ALL_PRINTHEAD_MOTION_COORDINATIONS):
    global X_MIN, X_MAX, Y_MIN, Y_MAX, Z_MIN, Z_MAX
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)

    # Define limits of the graph
    ax.set_xlim(X_MIN,X_MAX)
    ax.set_ylim(Y_MIN,Y_MAX)

    # Label the axis
    plt.xlabel("[mm]")
    plt.ylabel("[mm]")

    # Major ticks every 20, minor ticks every 5
    major_ticks = np.arange(X_MIN, X_MAX + 1, 20)
    minor_ticks = np.arange(Y_MIN, Y_MAX + 1, 5)
    ax.set_xticks(major_ticks)
    ax.set_xticks(minor_ticks, minor=True)
    ax.set_yticks(major_ticks)
    ax.set_yticks(minor_ticks, minor=True)

    # And a corresponding grid
    ax.grid(which='both')

    # Or if you want different settings for the grids:
    ax.grid(which='minor', alpha=0.2)
    ax.grid(which='major', alpha=0.5)
    plt.grid()
    for index in range(1,len(ALL_PRINTHEAD_MOTION_COORDINATIONS)):
      if (ALL_PRINTHEAD_MOTION_COORDINATIONS[index-1][3] == -1 and ALL_PRINTHEAD_MOTION_COORDINATIONS[index][3] == -1) or (ALL_PRINTHEAD_MOTION_COORDINATIONS[index - 1][3] == -2 and ALL_PRINTHEAD_MOTION_COORDINATIONS[index][3] == -1):
        xPoints = [ALL_PRINTHEAD_MOTION_COORDINATIONS[index - 1][0], ALL_PRINTHEAD_MOTION_COORDINATIONS[index][0]]
        yPoints = [ALL_PRINTHEAD_MOTION_COORDINATIONS[index - 1][1], ALL_PRINTHEAD_MOTION_COORDINATIONS[index][1]]
        plt.plot(xPoints, yPoints, "b--", linewidth=0.5)
      else:
        xPoints = [ALL_PRINTHEAD_MOTION_COORDINATIONS[index - 1][0], ALL_PRINTHEAD_MOTION_COORDINATIONS[index][0]]
        yPoints = [ALL_PRINTHEAD_MOTION_COORDINATIONS[index - 1][1], ALL_PRINTHEAD_MOTION_COORDINATIONS[index][1]]
        plt.plot(xPoints, yPoints, "r-", linewidth=1)
    plt.title("Expected Printhead Path and Printed Shape")
    plt.show()
```

```
def GetInfo():
    global X_CURRENT, Y_CURRENT, Z_CURRENT, X_ABS_MAX, Y_ABS_MAX, Z_ABS_MAX, X_ABS_MIN, Y_ABS_MIN, Z_ABS_MIN
    global x_origin_gap, y_origin_gap, z_origin_gap, x_upper_gap, y_upper_gap, z_upper_gap
    print("###################")
    print("x_origin_gap = ", x_origin_gap)
    print("y_origin_gap = ", y_origin_gap)
    print("z_origin_gap = ", z_origin_gap)
    print("x_upper_gap = ", x_upper_gap)
    print("y_upper_gap = ", y_upper_gap)
    print("z_upper_gap = ", z_upper_gap)
    print("###################")
    print("X_ABS_MIN = ", X_ABS_MIN)
    print("X_ABS_MAX = ", X_ABS_MAX)
    print("Y_ABS_MIN = ", Y_ABS_MIN)
    print("Y_ABS_MAX = ", Y_ABS_MAX)
    print("Z_ABS_MIN = ", Z_ABS_MIN)
    print("Z_ABS_MAX = ", Z_ABS_MAX)
    print("###################")
    print("X_MIN = ", X_MIN)
    print("X_MAX = ", X_MAX)
    print("Y_MIN = ", Y_MIN)
    print("Y_MAX = ", Y_MAX)
    print("Z_MIN = ", Z_MIN)
    print("Z_MAX = ", Z_MAX)

# Print current location with respect to the absolute origin
def GetCurrentLocationABS():
    global X_CURRENT, Y_CURRENT, Z_CURRENT
    print("Current Location with respect to absolute origin:")
    print("X: ", X_CURRENT)
    print("Y: ", Y_CURRENT)
    print("Z: ", Z_CURRENT)

# Print current location with respect to the redefined origin
def GetCurrentLocationREL():
    global X_CURRENT, Y_CURRENT, Z_CURRENT
    print("Current Location with respect to redefined origin:")
    print("X: ", X_CURRENT - X_ABS_MIN)
    print("Y: ", Y_CURRENT - Y_ABS_MIN)
    print("Z: ", Z_CURRENT - Z_ABS_MIN)

```

```
# Learn current printing mode, printing is on or off
def GetPrintingMode():
    global PRINTING_MODE
    if PRINTING_MODE == 1:
        print("Printing: ON")
    else:
        print("Printing: OFF")

# Do home operation
def HomeOperation():
    global ALL_PRINTHEAD_MOTION_COORDINATIONS
    print("##############################################")
    print("Home Operation is working")
    G28_Home()
    if PRINTING_MODE == 0:
      ALL_PRINTHEAD_MOTION_COORDINATIONS.append([X_CURRENT, Y_CURRENT, Z_CURRENT, -1])    #-1: printng off, -2: printing on
    else:
      ALL_PRINTHEAD_MOTION_COORDINATIONS.append([X_CURRENT, Y_CURRENT, Z_CURRENT, -2])    #-1: printng off, -2: printing on

def DrawPrintheadMotions():
    global ALL_PRINTHEAD_MOTION_COORDINATIONS
    print("##############################################")
    print("Expected path of the printhead and will be printed shape is working")
    DrawPrintheadMotionsHelper(ALL_PRINTHEAD_MOTION_COORDINATIONS)

# Convert and Download the gcode file
def FormTXTFileOfGCODE(filename):
  global G_CODE
  G_CODE = G_CODE[1:]     # This is because we have \n at the begenning if we do home operation
  filename = filename + ".gcode"     # we want to download as txt file
  with open(filename, 'w') as f:
      f.write(G_CODE)
  files.download(filename)
```

```python
##########################################################################################################################
# FUNCTIONS THAT USER CAN USE

# Follow a line, from last point to entered coordinates to target point that entered
# speedValue: feed rate of the printhead, speed of the printhead in mm/min
# X: x axis of target point in mm
# Y: y axis of target point in mm
# Z: z axis of target point in mm
def LinearMotionABS(speedValue, X = 0, Y = 0, Z = 0):
    global X_ABS_MIN, Y_ABS_MIN, Z_ABS_MIN, PRINTING_MODE
    boundaryCheck = BoundryCheck(X_ABS_MIN + X,Y_ABS_MIN + Y, Z_ABS_MIN+ Z)
    print("##############################################")
    print("Absolute Linear Motion is working to coordinates: X = " + str(X_ABS_MIN + X) + " Y = " + str(Y_ABS_MIN + Y) + " Z = " +  str(Z_ABS_MIN+ Z))
    if boundaryCheck:
        if PRINTING_MODE == 0:
          ALL_PRINTHEAD_MOTION_COORDINATIONS.append([X_ABS_MIN + X, Y_ABS_MIN + Y, Z_ABS_MIN+ Z, -1])    #-1: printng off, -2: printing on
        else:
          ALL_PRINTHEAD_MOTION_COORDINATIONS.append([X_ABS_MIN + X, Y_ABS_MIN + Y, Z_ABS_MIN+ Z, -2])    #-1: printng off, -2: printing on
        G0_ABS(speedValue, X_ABS_MIN + X, Y_ABS_MIN + Y, Z_ABS_MIN+ Z)
        #GetCurrentLocationABS()
        GetCurrentLocationREL()

# Follow a line, from last point to last point + entered coordinates
# speedValue: feed rate of the printhead, speed of the printhead in mm/min
# X: distance in x axis that wanted to move in mm
# Y: distance in y axis that wanted to move in mm
# Z: distance in z axis that wanted to move in mm
def LinearMotionREL(speedValue, X = 0, Y = 0, Z = 0):
    global X_CURRENT, Y_CURRENT, Z_CURRENT, X_ABS_MIN, Y_ABS_MIN, Z_ABS_MIN
    boundaryCheck = BoundryCheck(X_ABS_MIN + X + X_CURRENT, Y_ABS_MIN + Y + Y_CURRENT, Z_ABS_MIN + Z + Z_CURRENT)
    print("##############################################")
    print("Relative Linear Motion is working with distances: X = " + str(X_ABS_MIN + X + X_CURRENT) + " Y = " + str(Y_ABS_MIN + Y + Y_CURRENT) + " Z = " + str(Z_ABS_MIN + Z + Z_CURRENT))
    if boundaryCheck:
        if PRINTING_MODE == 0:
          ALL_PRINTHEAD_MOTION_COORDINATIONS.append([X_ABS_MIN + X + X_CURRENT, Y_ABS_MIN + Y + Y_CURRENT, Z_ABS_MIN + Z + Z_CURRENT, -1])    #-1: printng off, -2: printing on
        else:
          ALL_PRINTHEAD_MOTION_COORDINATIONS.append([X_ABS_MIN + X + X_CURRENT, Y_ABS_MIN + Y + Y_CURRENT, Z_ABS_MIN + Z + Z_CURRENT, -2])    #-1: printng off, -2: printing on
        G0_REL(speedValue,X,Y,Z)
        #GetCurrentLocationABS()
        GetCurrentLocationREL()

```

```python
# Turn on or off printing mode
# if printingMode = "ON": Turn on the printing
# if printingMode = "OFF": Turn on the printing
def PrintingOnOff(printingMode):
    global PRINTING_MODE
    if(printingMode == "ON"):
        PRINTING_MODE = 1
        M42(27,255,1)
    elif(printingMode == "OFF"):
        PRINTING_MODE = 0
        M42(27,0,1)
    else:
        print("ERROR: YOU ENTERED WRONG INPUT FOR PRINTING")
    print("##############################################")
    GetPrintingMode()
```

Figure 12: Written code and functions in google collab

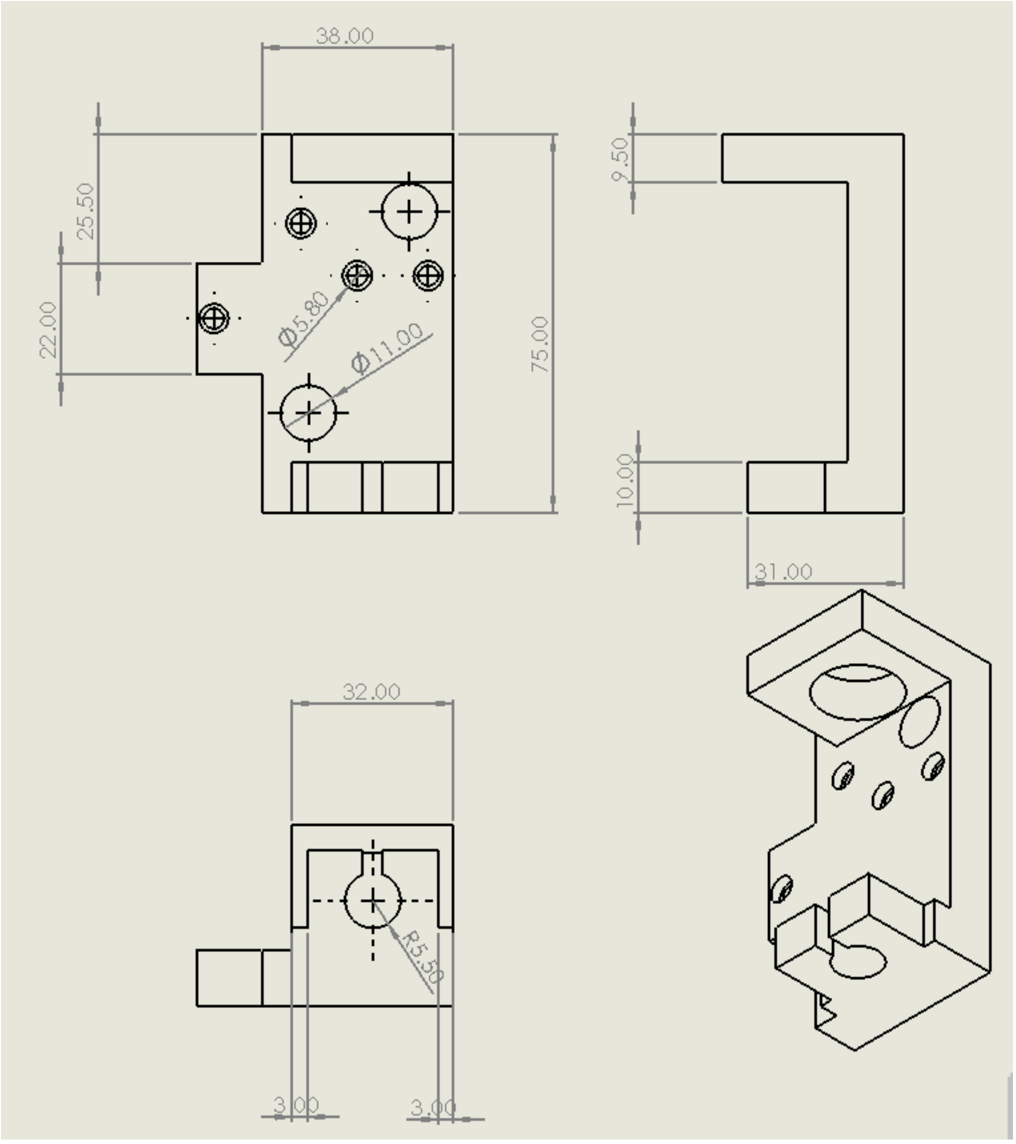
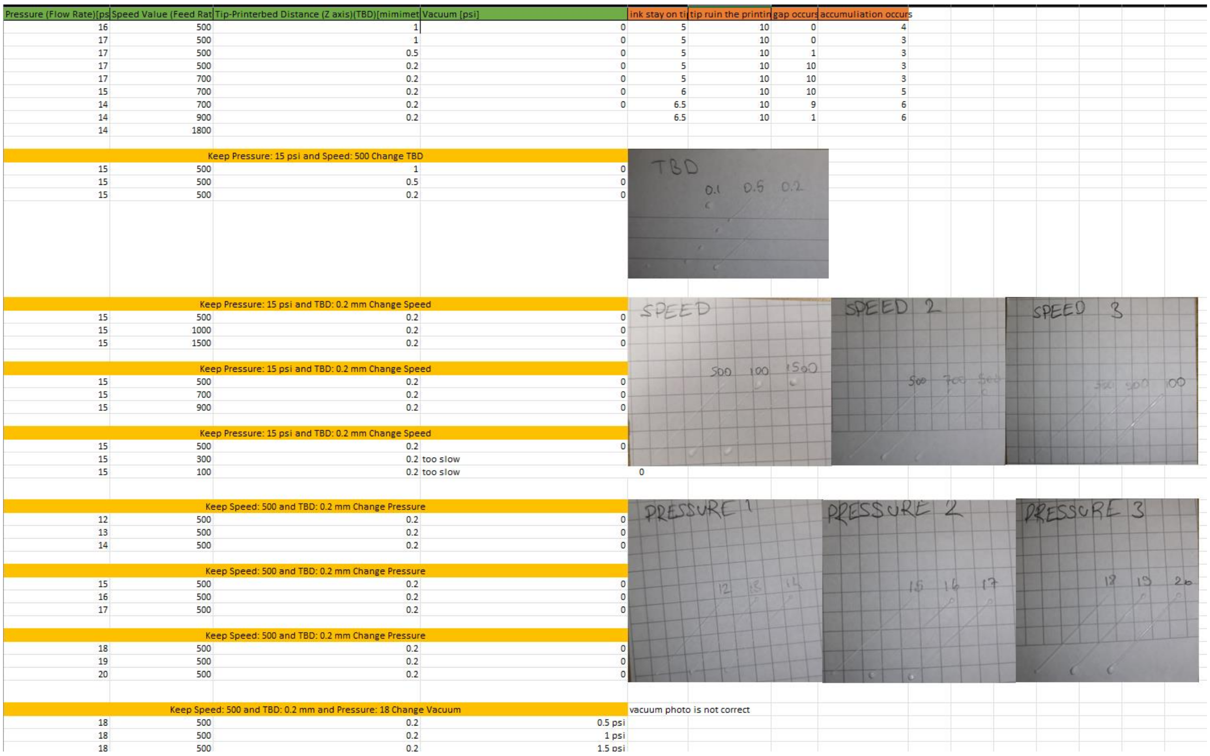


Figure 13: Technical Drawing of the Holder



Pressure (Flow Rate)[ps	Speed Value (Feed Rat	Tip-Printerbed Distance (Z axis)(TBD)[mimimet	Vacuum [psi]	ink stay on ti	tip ruin the printin	gap occurs	accumuliation occurs
16	500	1	0	5	10	0	4
17	500	1	0	5	10	0	3
17	500	0.5	0	5	10	1	3
17	500	0.2	0	5	10	10	3
17	700	0.2	0	5	10	10	3
15	700	0.2	0	6	10	10	5
14	700	0.2	0	6.5	10	9	6
14	900	0.2		6.5	10	1	6
14	1800						
Keep Pressure: 15 psi and Speed: 500 Change TBD							
15	500	1	0				
15	500	0.5	0				
15	500	0.2	0				
Keep Pressure: 15 psi and TBD: 0.2 mm Change Speed							
15	500	0.2	0				
15	1000	0.2	0				
15	1500	0.2	0				
Keep Pressure: 15 psi and TBD: 0.2 mm Change Speed							
15	500	0.2	0				
15	700	0.2	0				
15	900	0.2	0				
Keep Pressure: 15 psi and TBD: 0.2 mm Change Speed							
15	500	0.2	0				
15	300	0.2	too slow				
15	100	0.2	too slow	0			
Keep Speed: 500 and TBD: 0.2 mm Change Pressure							
12	500	0.2	0				
13	500	0.2	0				
14	500	0.2	0				
Keep Speed: 500 and TBD: 0.2 mm Change Pressure							
15	500	0.2	0				
16	500	0.2	0				
17	500	0.2	0				
Keep Speed: 500 and TBD: 0.2 mm Change Pressure							
18	500	0.2	0				
19	500	0.2	0				
20	500	0.2	0				
Keep Speed: 500 and TBD: 0.2 mm and Pressure: 18 Change Vacuum				vacuum photo is not correct			
18	500	0.2	0.5 psi				
18	500	0.2	1 psi				
18	500	0.2	1.5 psi				

Figure 14: Excel Table about calibration of 3D printer

Figure 15: Successful printing