Activating a CMOS Pixelated Capacitive Sensor Platform by Inkjet Printer - Software

Bachelor Graduation Thesis

by

E. Tuinstra & T. A. Tommel

in collaboration with

N. Chen
D. van Krieken
D. Rueda Lindemann
S. Verweij

Monday 8th July, 2024

Instructors:	Dr. S. Kundu
	Prof. dr. ir. F. P. Widdershoven
Teaching assistant:	T. Shen
Project duration:	April, 2024 – June, 2024
Faculty:	Faculty of Electrical Engineering, Mathematics & Computer Science, Delft

TU Delft

Abstract

This thesis explores the modification of an EPSON EcoTank ET-8500 desktop inkjet printer to facilitate printing on CMOS chips for the cost-effective production of capacitive sensors. By reengineering both the hardware and software of the ET-8500, we aim to repurpose this standard desktop printer into a tool capable of advanced, precision printing for electronic applications.

The report details the development process, beginning with an overview of printer drivers and the Epson Standard Code for Printers (ESC/P) language. Utilising this foundation, custom software was developed in Python, featuring a user-friendly interface for seamless operation. The customised system was rigorously tested to assess the modified printer's capabilities, with results thoroughly analysed to determine feasibility and performance. The thesis concludes with a discussion of findings, along with recommendations for future enhancements and applications.

Preface

This report marks the completion of the Bachelor's degree in Electrical Engineering, a journey that has been both challenging and rewarding.

This project has allowed us to combine the knowledge obtained throughout the study into one piece. It involves all steps from design to implementation and eventually testing. The project also included creating a business plan and thinking about the ethical complications. This work has been handed in separately.

The project has also strengthened the collaboration with different teams as the subgroups; hardware, software and measurements needed to work closely together to come to a final system working as one.

We would also like to thank our supervisors Frans Widdershoven and Suman Kundu for their guidance. And our teaching assistant Tao Shen for helping throughout the project. In particular, we want to thank Suman Kundu, he has been a great inspiration and help, without whom we may never have reached the product we have now. He always helped us in finding solutions when we found a roadblock and stimulated us to be creative in all situations or to at least *think about it*.

E. Tuinstra & T. A. Tommel
Delft, July 2024

Contents

1

Introduction

1.1 Inkjet Printers

Nowadays printing technology has become widespread, and available to most if not all middle-class families. Despite its rather low costs, the technology is already very mature and can print to a very precise degree. The printer used for research in this thesis, the Epson ET-8500, has shown from testing that it is capable of printing with an incredible accuracy of up to a tenth of a millimetre distance between two different droplets with ink droplets with a diameter of around 80 micrometres.

With this accuracy, a lot more can be achieved than just daily printing jobs such as high-precision deposits of fluids. If one were to be able to print on CMOS chips, then it would enable both in theory and in practice the fabrication of capacitive sensors readily available for anyone to do at home. This would make sensor technology a lot more accessible as compared to before, for one needed very expensive and complicated machinery to produce them. This in turn makes the product also rather expensive.

1.2 State of the Art

The Epson ET-8500 is an inkjet printer that uses a piezoelectric printing technology. This technology is currently the most reliable in terms of drop consistency and speed. It uses a piezo-driven inkjet printhead that is activated by a voltage waveform applied to a piezoelectric transducer. This in turn changes in shape when activated, generating a pressure pulse in the ink chamber, causing a droplet of the printing substance to drop[1].

Although the technology is readily available, the option for a consumer to easily print any substance on any substrate is far from accessible. This is due to a few different reasons. Two of them being the nature of the printers usually only allowing for (easy) printing on paper, thus the transition to other substrates requires significant alterations to the printer and the second is that printing software is not properly documented for the public to view and ultimately to understand.

Despite the fact that manipulating a printer to do what a consumer exactly wants is made quite difficult by the manufacturers, there have been more people who have tried to control

their printer through unconventional ways. Such ways can mean both the making of significant hardware changes as well as the use of driver commands. More on this will be mentioned in section 1.3.

On the other hand, if one were to refer to more advanced printers, which are usually known as material printers. However, these do cross over more in the territory of 3D printers. As mentioned before, these printers can be rather pricey, with prices starting in the ten-thousands going up to over a million euros, but they can also be rather large, smaller ones are about double the size of household printers while the larger material printers can be as large as a medium-sized room, which also depends on the kind of material that is printed by the printer.

Theoretically, these more expensive printers are also able to print on a resolution of up to 1200 dpi up to an accuracy of even 2900 dpi. According to the documentation of the Epson ET-8500, it should have a printing resolution of up to 5760 by 1440 dpi [2]. Although, in practice, the highest practical resolution is 720 x 720. Though, taking into account the sizing of the printed dots, this is plenty for the manufacturing of capacitive CMOS sensors.

1.3 Similar Research

As previously mentioned, this project is not entirely unique in its premise of taking more or full control of a household printer. In some cases an effort was also made to print on surfaces of one's own choice [3].

Another great source of information about bypassing the driver is an article about the production of radioactive phantoms [4] [5]. In this article a standard inkjet printer is used, unfortunately, the software they wrote could not be found and contacting the authors was unsuccessful.

Also research about the physics of print has been done [6]. This application describes the use of inkjet printing of light-emitting polymer displays. This article goes in-depth about how the different factors affect accuracy. Important factors are; the spreading of the liquid, vaporisation of the liquid and falling of the droplets.

1.4 Thesis Outline

The thesis is structured as follows. In chapter 2 the problem definition is given and a program of requirements is created. In chapter 3 will be explained how the printer works and how to control it. Using this information, chapter 4 will elaborate on how this is implemented in the software. Using the created software, chapter 5 will describe how it is tested and how problems were solved. In chapter 6 the resulting printing quality is shown. At last, chapter 7 gives a conclusion together with future work.

2

Project Description

2.1 Problem Definition

To increase the accessibility of printing on chips to create sensors and to make it less expensive the Epson ET-8500 can be used. To achieve this some modifications need to be done, such as modifying the printer itself to allow for printing on the chips physically. Also, the digital print method needs to be changed to allow for dropping individual dots on the specified locations. Afterwards, the chips must be measured and analysed to see if the printing succeeded. These tasks are divided into three subgroups

- Hardware
- Software
- Measurements

This thesis focuses on the software part of the project, as this is the software subgroup. In Figure 2.1 is illustrated how the whole process of printing on the chip and measuring the result works. In red is outlined which part is handled by the software subgroup.

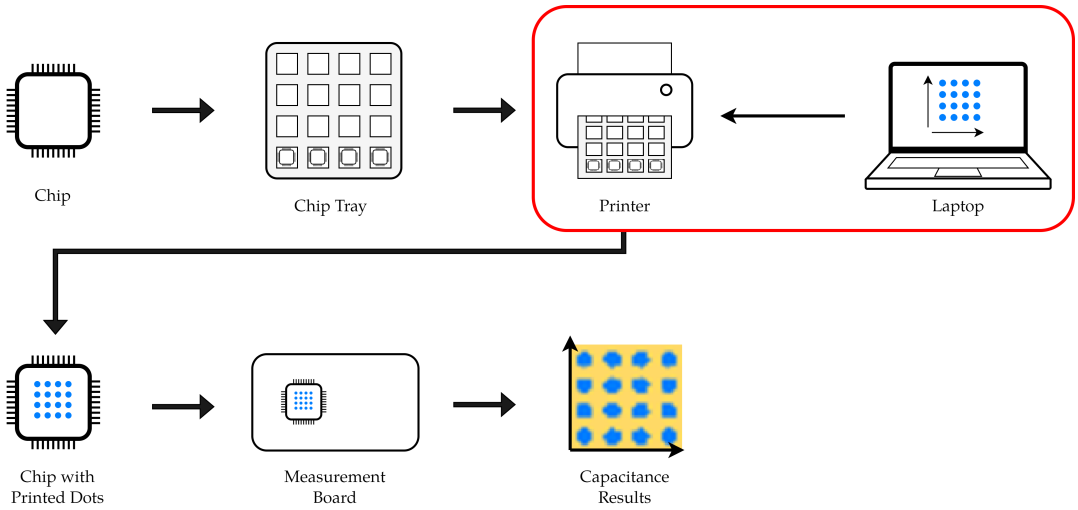


Figure 2.1: Block diagram of the whole process for printing and measuring a CMOS chip. Outlined in red is the part the software subgroup is responsible for.

The general tasks given to the software group are:

1. Hacking the printer driver or working around its limitations.
2. Designing printing patterns and algorithms to detect them in measured signals.
3. Multiple inks need to be printed on a single chip.
4. Develop a user-friendly software interface to use the modified printer platform.

This results in the problem definition:

"Can we develop a user-friendly interface that allows users to print individual dots with multiple inks using the Epson ET-8500?"

2.2 Programme of Requirements

Using the tasks a program of requirements is created. These requirements are split into mandatory requirements and trade-off requirements.

Mandatory Requirements or Constraints
These are requirements that the product has to comply with.

[1.1] The product will consist of a piece of software.

[1.2] The software must allow the user to create printing patterns with the use of a graphical user interface.

[1.3] The software must allow the user to print from a computer connected to the printer using a USB cable.

[1.4] The software must be installable on a Windows or Linux system.

[1.5] The software must allow the user to print to the CD tray of the ET-8500 (which will be modified to hold CMOS chips).

[1.6] The software must allow the user to specify the locations of the printed dots.

[1.7] The software must allow the user to control individual nozzles in the printhead of the printer.

[1.8] The software must allow the user to set the colour of the individual dots to all of the available inks.

[1.9] The software must have a manual describing all its functionalities.

[1.10] The diameter of the smallest dot is less than 80 μm [1].

[1.11] The closest spacing of the dots is less than 141 μm [2].

[1.12] It needs to be possible to print multiple dots on the exact same position.

[1]The biggest sensor array has a size of 418.3 μm by 357.1 μm [Figure A.1]. To be able to print 4 dots on the array horizontally, the dot size must be a minimum of 357.1 / 4 $\approx$ 80 μm, however this does not yet take into account the spacing of the dots.

[2]The spacing of the nozzles is 180 DPI $\approx$ 141 μm, this spacing should be the minimum achievable spacing.

Trade-off Requirements or Objectives

These are the trade-off requirements, the better the results the more satisfied the requirements are.

[2.1] The minimum dot size should be as small as possible.

[2.2] The minimum dot spacing should be as close as possible.

[2.3] The accuracy of the dot placement in a grid should be maximized.

3

Interfacing with the Epson ET-8500

3.1 Introduction

Under normal use of a desktop inkjet printer, the spooler is responsible for creating a job for the printer with the to-be-printed file. The spooler manages a queue of print jobs and allows the user to abort a print job. In this process, the spooler uses the printer driver to generate a sequence of bytes the printer understands to print. It does so by rasterising the file and creating a list of commands. For the user, this behaves as a black box, they only have to press print and the rest is handled for them.

The generated command sequences are used to control a printhead that races across the paper. In this printhead, a lot of nozzles are placed and droplets are combined to create the right colours. Trying to print individual droplets using this standard procedure is impossible because the printer attempts to ensure high quality and dispenses more droplets to ensure the pixel is visible. This method does not allow of control over the individual dispensing of the nozzles. To allow the possibility of setting individual dots the driver of the printer needs to be bypassed, this will be explained in the next section.

3.2 Printer Drivers

As mentioned in the previous section, the printer driver is responsible for converting a file that needs to be printed into a sequence of commands the printer can understand.

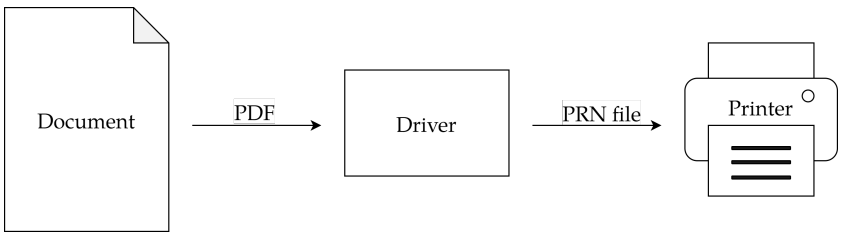


Figure 3.1: The normal procedure of printing a PDF file to a printer

In Figure 3.1 a block diagram of the normal printing process is shown. It can be seen how the driver transforms the PDF file into a PRN file, which is a file containing binary content. The

goal is to understand this file the driver creates and make it possible to generate this file using custom code.

The Epson ET-8500 supports multiple different drivers which all convert the file to different languages the printer understands. For this, Epson also created its own language called Epson Standard Code for Printers (ESC/P). This language has multiple variants, namely:

- **ESC/P**, the original version of the printer control language.
- **ESC/P2**, a more up to date version of ESC/P, it is backwards compatible with ESC/P, but adds more commands.
- **ESC/P-R**, the newer version of the old ESC/P2 language. This is supported by most new Epson printers.
- **ESC/P-R2**, again a more up to date version of the older version, ESC/P-R.

The documentation about the newer ESC/P-R and ESC/P-R2 is scarce and thus they are hard to decipher. For the older ESC/P2 language a reference manual from Epson exists [7]. For ESC/P-R the source code of the driver exists [8], which could be used for reference, but this would still be a hard time to decode. Therefore the ESC/P2 language is chosen. The next section will explain the printing language in further detail.

3.3 ESC/P2 Commands

The PRN file sent to the printer is a big binary file in which all commands are placed consecutively. All these commands are separated by a `ESC` ASCII character which is `0x1B` in hexadecimal notation [1].

To get a better understanding of the commands that are available and how they are use, the open-source driver Gutenprint has been used [9]. This is a collection of drivers that can be used on Linux by CUPS [10] to print. Gutenprint also provides documentation for developers on how the drivers work, which is useful information. It also allows for printing to a file instead of sending the commands to the printer. This makes it possible to look at the bytes normally sent to the printer. Using this method, an input file with a line can be sent, and the output can be compared to get an understanding of how the line is translated.

Using the Epson programming guide and the additional information from Gutenprint a lot of the commands could be decoded. To illustrate how one command looks, an example is given for the `SetRelativeVerticalPrintPosition` command.

Table 3.1: The documentation for the SetRelativeVerticalPrintPostition command, partially obtained from [7]

ESC (v nL nH m1 m2 m3 m4	
Name	Set Relative Vertical Print Position
Format	0x1B, 0x28, 0x76, nL, nH, m1, m2, m3, m4
Range	nL=0x04, nH=0x00 $0 \leq$ (m4 * 0x1000000 + m3 * 0x10000 + m2 * 0x100 + m1) * 1440 $\leq$ 0x1FFFFFFF

[1] In this thesis hexadecimal notation is used multiple times. To denote this, the convention of 0x is used as a prefix.

Description This command sets the vertical position of the printhead relative to the previous position. The printing medium (paper, CD, etc.) moves down by m vertical units. Here m = (m4 * 0x1000000 + m3 * 0x10000 + m2 * 0x100 + m1) and the vertical unit is specified by the SetUnit command, with a limit of 1440 per inch. So this means that in theory, the closest vertical spacing equals 1 / 1440 inches $\approx 18\,\mu m$.

In Table 3.1 the documentation is given in the same way the reference manual depicts it. The command begins with ESC, (, v, this is followed by nL and nH, these two form a 16-bit number representing the number of bytes following the command. In this case, this should always be 4 bytes. These bytes are for the units of position the paper should move and can be calculated as described in the description.

To print something all the different commands must be combined in the right order. In Table 3.2 the standard structure is shown.

Table 3.2: Sequence of commands for printing raster data

Transfer cycle	Description	Commands used
By document	Reset printer	Exit Remote Mode Exit Packet Mode Initialise Printer
	Initialise settings	Initialise Printer Enter remote mode Paper Feed Setup Set Media Information Exit Remote Mode
	Printing method control	Selects Graphics Mode Set Unit Monochrome or Color Mode Selection Select MicroWeave Printing Mode Turn Unidirectional Mode On/Off Selects Dot Size Set Raster Image Resolution
	Set print format	Set Page Format Set Pape Length Set Paper Dimension Set Print Method ID
By raster	Print the raster data	Set Vertical Print Position Set Horizontal Print Position Transfer Raster Image
By page	Eject the page	Form feed
By document	Terminate printing	Initialise printer Enter Remote mode Load NVR settings Job end Exit Remote mode

The first column tells in which transfer cycle the group of commands is used. *"By document"*

means that it is only used once in the prn file sent. This consists of commands for initialising settings for the printer to use such as the print media information and the dot size to use. The *"by raster"* are the commands used for every raster it prints. It first tells the position and then transfers the raster data. The next sections will explain how the dots can be printed in the correct position with the correct sizes and colours.

3.4 Printing Rasters

To understand how the rasters are printed, it is important to take a look at the printhead. In Figure 3.3 a diagram of the nozzles of the printhead is visible. Every colour has its own column. Each column consists of 180 nozzles and is one inch long. This results in a maximum vertical dots per inch (DPI) of 180. This effect can be surpassed by moving the paper in the paper feed direction by 1/360 inch, resulting in the nozzles being exactly between the original positions as portrayed in Figure 3.2. This makes it possible to print in higher DPIs than the nozzle spacing itself allows.

Another thing to note is that photo black, magenta and gray are offset by 1/360 inch in the vertical direction as seen in Figure 3.3. This needs to be accounted for when printing with these colours.

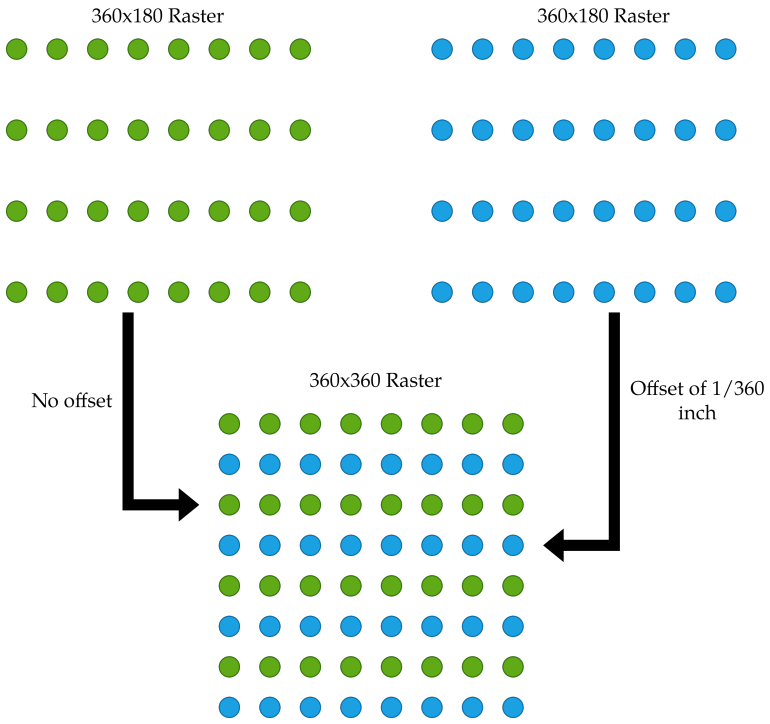


Figure 3.2: The figure shows two rasters with a horizontal resolution of 360 DPI and a limited vertical resolution of 180 DPI. The figure illustrates that a vertical resolution of 360 DPI can be obtained by applying an offset of 1/360 inches to one of the rasters.

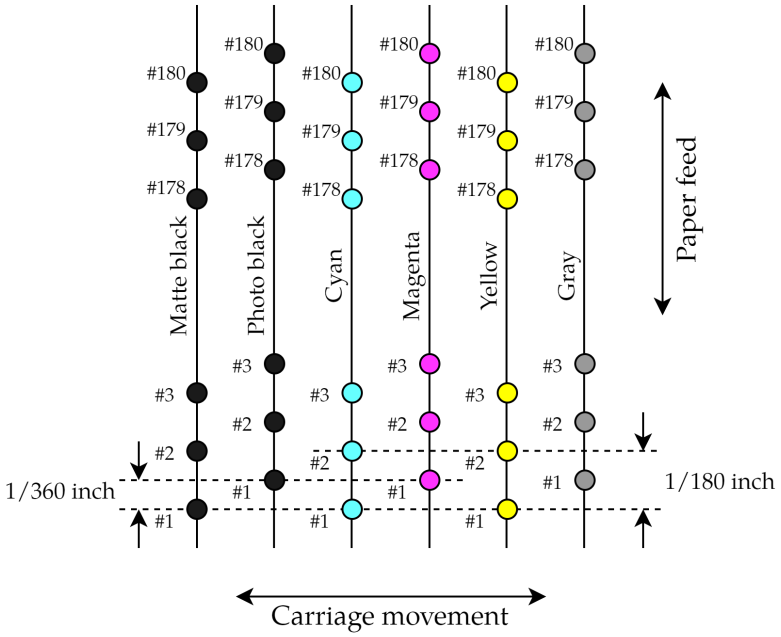


Figure 3.3: Printhead nozzle diagram, partially obtained from [7]

The next subsections will explain how the following goals are achieved:

- Specifying the location of the raster.
- Changing the sizing of the dots.
- Printing with different colours.
- Setting the DPI of the raster.

Location of the Raster

To specify the location of the raster the `SetAbsoluteHorizontalPrintPosition` and `SetRelativeVerticalPrintPosition` are used. The location can be specified in units that are assigned by the `SetUnit` command. In Figure 3.4 the positioning coordinate system is depicted. The origin of the page management is on the top left and the origin of the position management is offset by the margins. In the software implementation, which is explained in chapter 4, the margins are left zero, so that the origin of the page management is the same as of the position management.

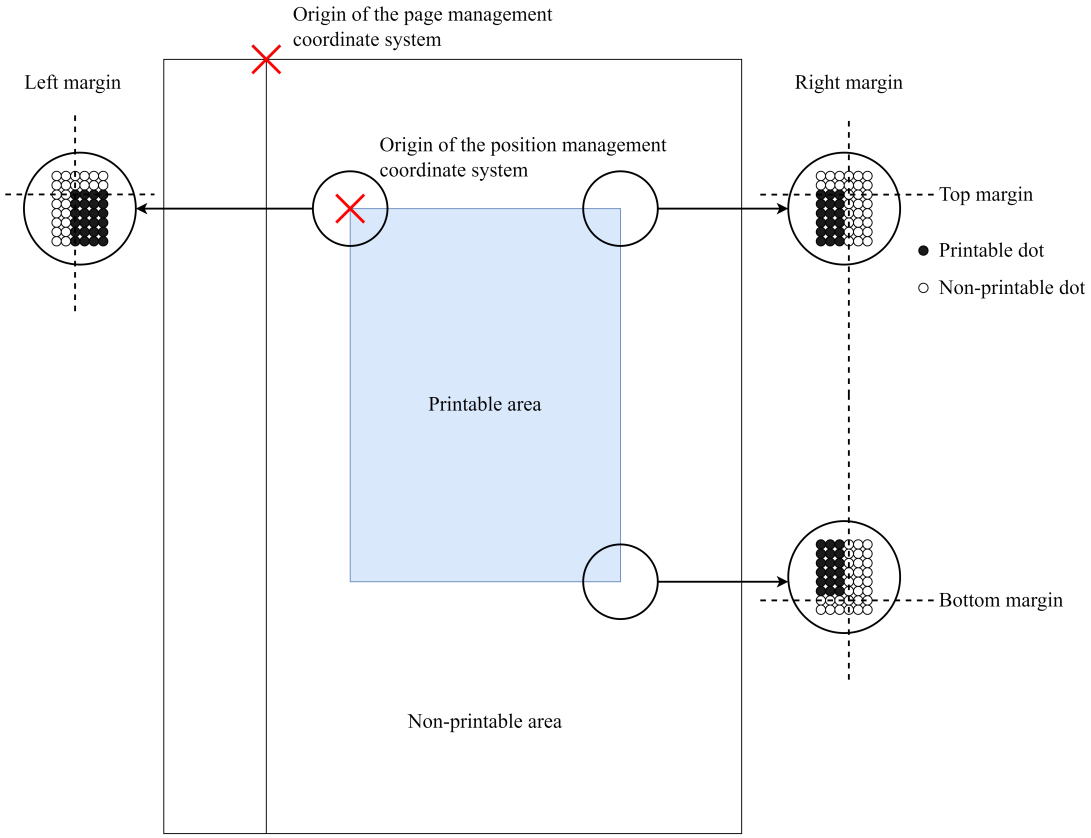


Figure 3.4: Page management diagram, partially obtained from [7]

These positioning methods are needed to solve requirements [1.6] and [1.12].

Size of the Dots

To fulfil requirement [2.1] a way for controlling the dot size is needed. The size of the dots is specified with the `SelectDotSize` command. This command has three known possibilities:

- VSD1 2-bit = 0x11, the biggest size dot
- VSD2 2-bit = 0x12, the medium size dot
- VSD3 2-bit = 0x13, the smallest size dot

VSD stands for Variable-Sized Droplet Technology. This is a technique where by varying the voltage that drives the piezoelectric elements, a nozzle is capable of dispensing different droplet sizes.

The setting is specified by a hexadecimal number, which represents the size, thus it could be possible that another number corresponds to different sizes, but these are unknown.

Where the `SelectDotSize` command is an easy way to specify the size of all the dots in a raster, there is also the option to specify some relative variation of the dot sizes within the raster. Every dot in the `TransferRasterImage` command is given by two bits specifying its relative size. This means that there are four possibilities:

- 00_2 means no dot
- 01_2 means a small dot

- 10_2 means a medium size dot
- 11_2 means a large dot

Colour of the Dots

To address requirement [1.8], it must be possible to print in different colours. To change the colour of the dots, the `TransferRasterImage` itself needs to be used. Besides the data for the raster it is going to print it also needs to colour with which it will print the raster.

Spacing (DPI) of the Raster

Requirements [1.6] and [1.7] arise the need for defining the resolution of the raster being printed. Using the command `SetRasterImageResolution`, the resolution of the following `TransferRasterImage` commands is determined. This resolution can be 180 DPI vertical and 180, 360 or 720 DPI horizontal.

4

Software Implementation

4.1 Introduction

The previous chapter explained how can be communicated with the Epson printer. It would be useful to have an application that is able to construct the correct commands for the printer and makes it easy to modify the parameters. This chapter will explain how the *graphic user interface* or GUI works and how it is created.

The source code can be found at `https://github.com/Thomas-164/ESCP2-Client` [11]. The manual for the application is also placed in this repository in the folder *manual*.

4.2 GUI

A GUI is used to create user-friendly applications. This application in particular makes it easy for an end-user to modify parameters and print the created commands. The GUI is created in Python [12]. Python is used because it is very accessible and easy for people to understand and adopt. To create the GUI, the library PyQt5 [13] is used. This is based on Qt, which is a powerful, cross-platform GUI library. This allows for running the GUI on different operating systems. This is needed to meet requirement [1.4].

To break down the options of the GUI, the layout can be separated into multiple tabs, each tab is responsible for its own set of features and will be explained in the next subsections.

Print Tab

The print tab is the default open tab when the application launches. In Figure 4.1 the view is shown. This tab can be further divided into four panels.

Preview Panel The top left panel, on this panel a preview of the dots that are going to be printed can be seen. This preview is created by traversing all commands and plotting them respectively. The preview panel can also display the tray and the locations of the chips. This can be seen in Figure 4.2.

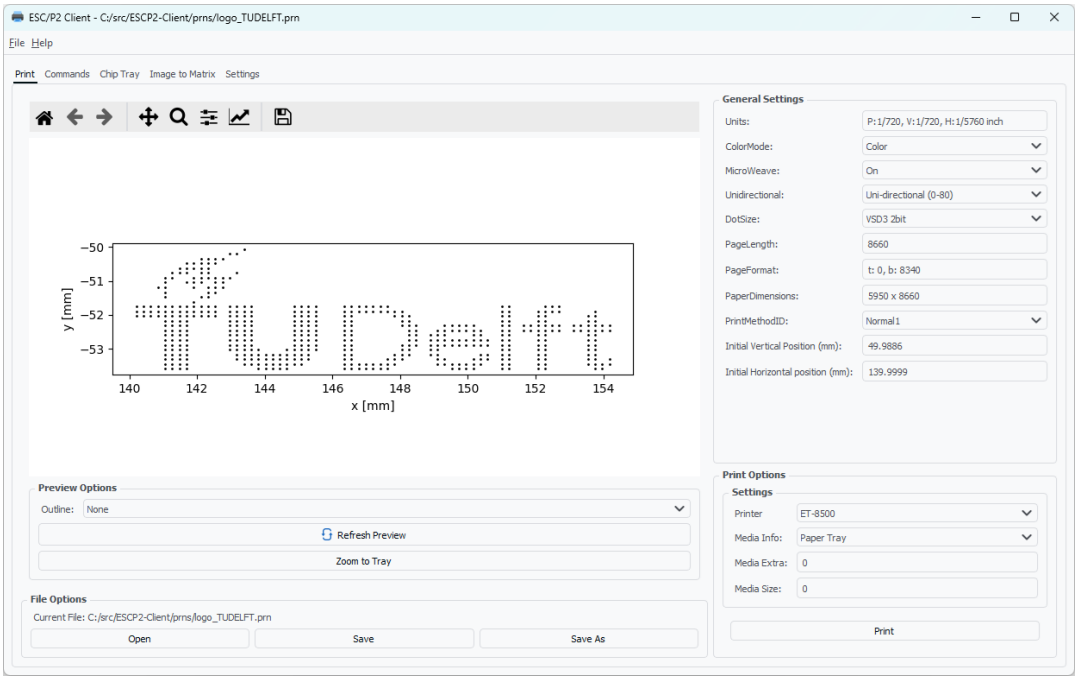


Figure 4.1: ESC/P2-Client print tab view

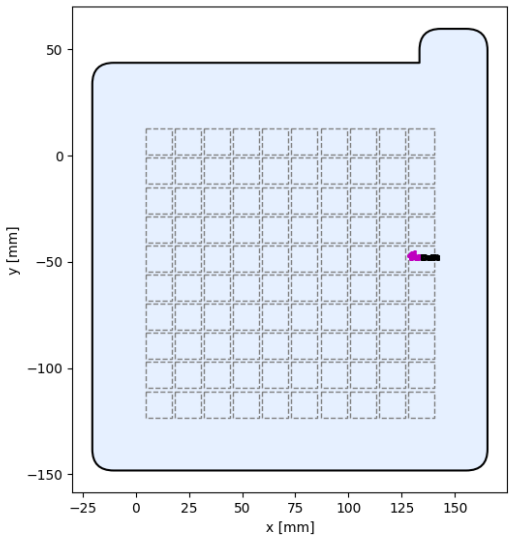


Figure 4.2: The preview panel with the chip tray on the background. The TU Delft logo is printed on the 10th column 5th row. With TU in magenta and Delft in photo black.

File Options The bottom left panel. This panel is used to display which file is currently being edited (which can also be seen in the application's title). Furthermore, it has the standard options of saving and opening a new file. These options are also available under the file button in the menubar. These also have shortcuts to make the experience even faster.

General Settings The top right panel. The general settings convey settings that apply to the entire document that will be printed. The following options can be seen:

- Units: These are the units used for the positioning of rasters.
- ColorMode: The mode in which the printer prints colours.
- MicroWeave: Select if MicroWeave is turned on or off.
- Unidirectional: If the printer prints in only one direction or not.
- DotSize: The size of the dots.
- PageLength: The length of the page.
- PageFormat: Used for specifying the top and bottom margins.
- PaperDimensions: The dimension of the paper.
- PrintMethodID: The method ID used for printing. This is a 2-byte hex code.
- Initial Vertical Position: The vertical position of where the printing will start.
- Initial Horizontal Position: The horizontal position of where the printing will start.

Print Options The bottom right panel. This panel is used for actually printing the created pattern. It consists of a dropdown with available printers and the options to select the media you want to print on. This can either be on paper or the CD tray. This is needed to fulfil requirement [1.5]. In addition, it has the option to specify extra media info. This is a byte which represents the type of media it is going to print. It is not entirely clear what this does, but it can alter the printing technique used to adapt to that specific media. For example, setting *Media Extra* to 91 results in the printer printing more slowly.

Commands Tab

The commands tab gives the most control over what is going to be printed, as it allows the user to shuffle the commands around and modify them individually. On the left, the options for manipulating the order of the commands are visible. When a command is selected options for editing can be seen on the right. For every command, the raw bytes can be directly changed. For some more special commands, custom widgets are added to make it more user-friendly. Examples of this are: `SetRelativeVerticalPrintPosition`, `SetAbsoluteHorizontalPrintPosition`, `TransferRasterImage` and `SetRasterImageResolution`.

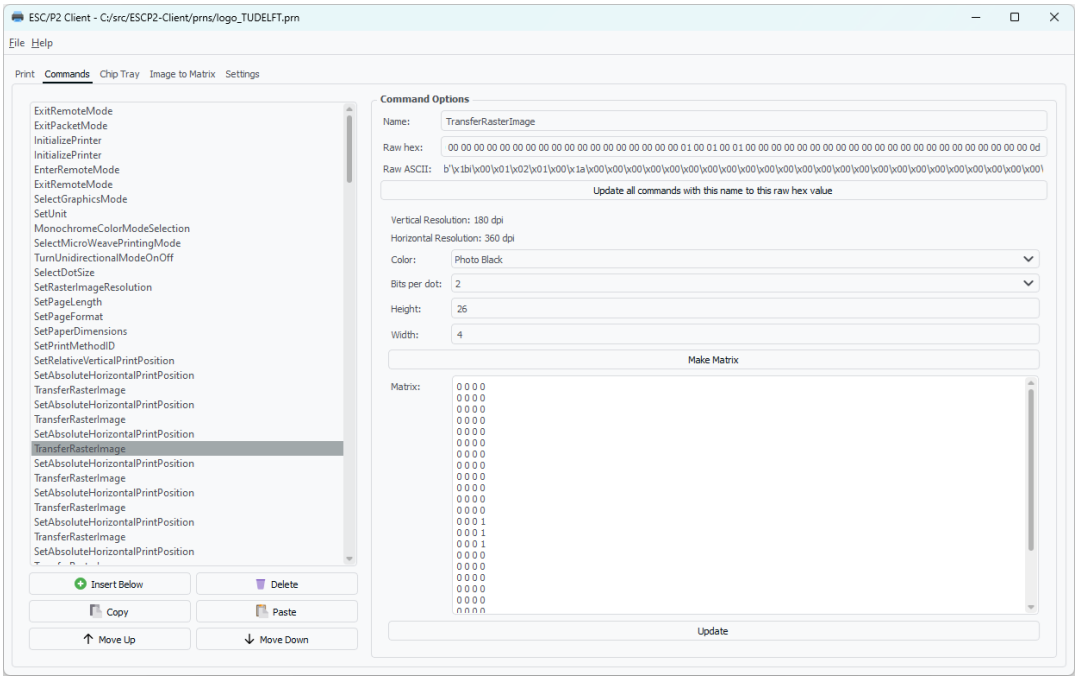


Figure 4.3: ESC/P2-Client commands tab view

The Figure 4.3 shows in particular how the `TransferRasterImage` command looks. It can be seen how the matrix can be edited to modify the raster that is going to be printed. This is implemented to ensure requirement [1.2] is met.

Chip Tray Tab

The chip tray tab is used to transfer the design pattern to all the locations of the chips. In Figure 4.4 the view can be seen. By selecting the chips, in this image chips A1, B1, C1 and D1 are selected, and pressing on *"Convert to commands"*, the pattern will automatically be transferred to the locations of the selected chips. This allows for easily printing the same design on multiple chips.

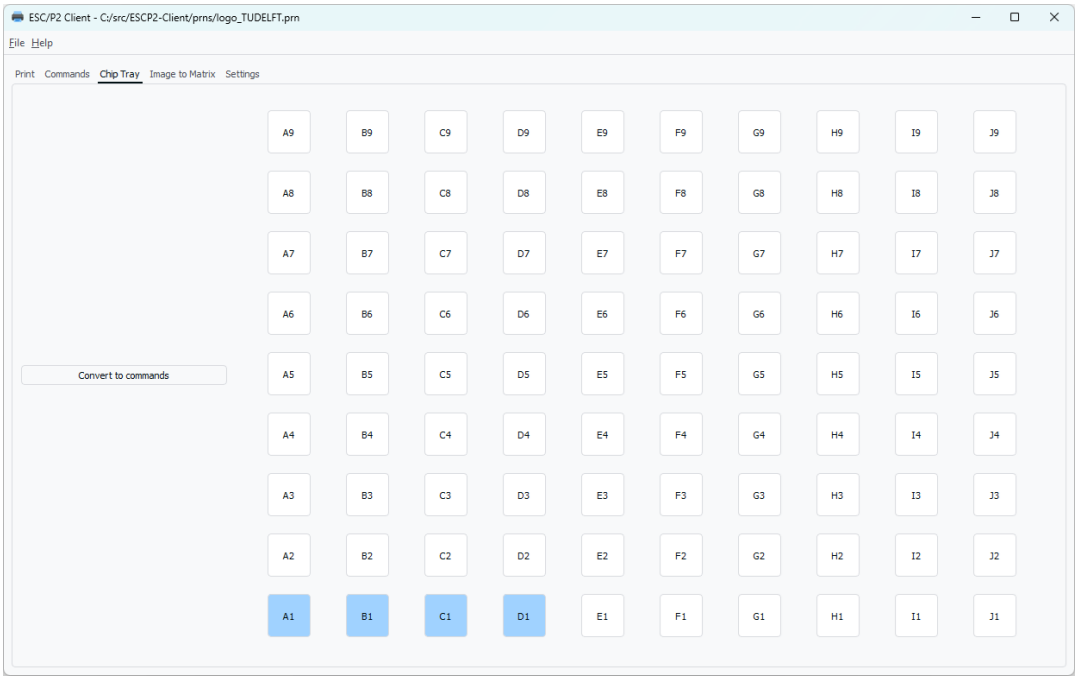


Figure 4.4: ESC/P2-Client chip tray tab view

Image to Matrix Tab

All raster data sent to the printer is a matrix of ones and zeros. To make it easier to create these matrices, the Image to Matrix tab is created. With this tab, an image can be selected and a matrix with specified width and height is generated. The threshold controls which pixels result in a one and which ones to a zero. This matrix data can then be copied to a `TransferRasterImage` command.

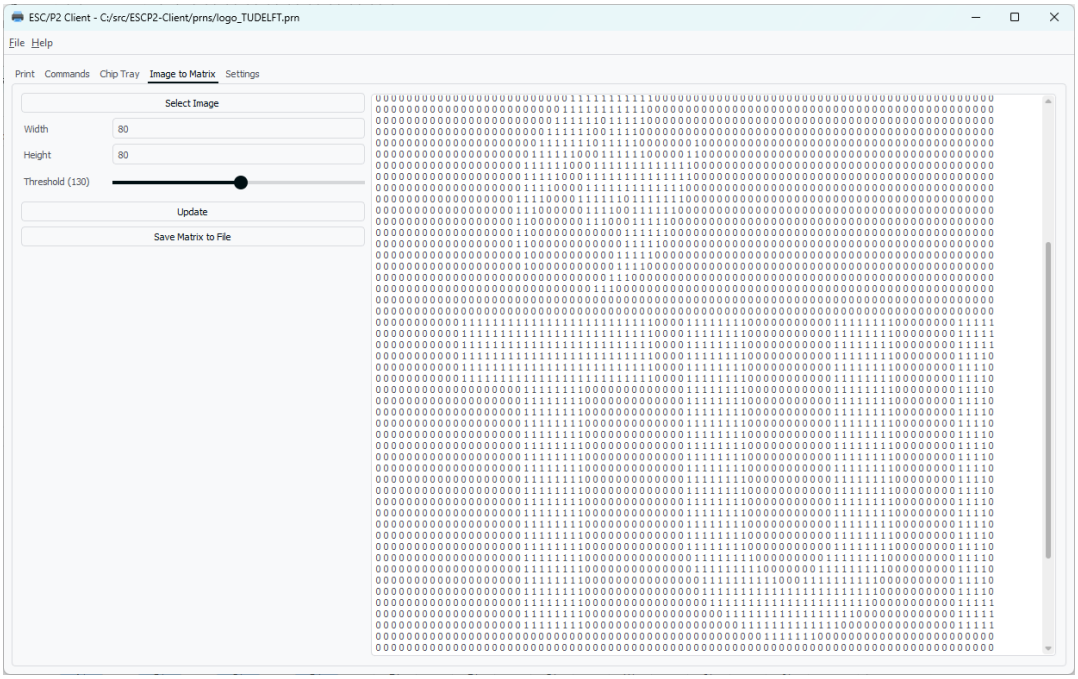


Figure 4.5: ESC/P2-Client image to matrix tab view

4.3 Code Quality

It is not feasible to explain every detail of the software, therefore only the interesting aspects of the code are explained.

Objection Orientated Programming

To keep the code organised object orientated programming is used [14]. In Listing 4.1 the code of the abstract base class `EscCommand` is visible. All commands inherit from this base class and therefore must implement the abstract methods. This helps create consistency across all commands.

Listing 4.1: Python code of the abstract base class for the EscCommand

```
class EscCommand(ABC):
    """
    Abstract class for ESC commands.
    """
    def __init__(self, data: bytes):
        self.data = data

    @abstractmethod
    def decode(self):
        """
        Decode the data.
        """
        pass

    @classmethod
    @abstractmethod
    def default(cls):
        """
        Returns the default command.
        """
        return cls(b'')

    def __str__(self):
        return f'{self.__class__.__name__}'
```

This allows for storing a list of `EscCommand`'s on which all the same methods can be called.

Use of GitHub

To keep track of changes and make it easier to collaborate with multiple people on the code, GitHub [15] is used. GitHub is version control software that saves all changes through the use of commits. This allows for working in separate branches for new features. It also helps to implement continuous integration (CI), which allows for automatic testing when a new release is created. More on testing in section 4.4.

Architecture: Separation of Business Logic and User Interface

The program is split into two modules, namely:

- **escp2**: This module is responsible for understanding the commands and updating the correct bytes. It also has the functionality to read in and create a prn file.
- **gui**: This module contains all code for the visualisation of the application.

By splitting the business logic and the user interface, the maintainable increases because changes in one area like business logic don't affect the UI design, making updates and debugging eas-

ier and more isolated. It also has a big advantage for testing, because the logic can be testing without needing the user interface, this is further explained in the next section.

4.4 Testing

As explained in subsection 4.3.2, GitHub is used which allows for continuous integration. Every time new code is pushed these tests are automatically run to check if everything still works as expected. These tests are so-called unit tests. They assert that the expected outcome of a function equals the actual outcome given certain test data. In Table 4.1 the percentage of covered code can be seen. This only includes the code responsible for the creation of the commands and not the GUI itself.

Table 4.1: Test coverage results of the tests

Name	Statements	Miss	Coverage [1]
escp2_client/utils.py	81	14	83 %
escp2_client/escp2/commands.py	383	154	60 %
escp2_client/escp2/commands_utils.py	136	23	83 %
Total	600	191	68 %

The reason for the `escp2_client/escp2/commands.py` to be so long is that every command that exists for the printer is in there. This could all be divided into separate files but would result in many files.

4.5 Software Manual

In accommodation with the software also a manual is created. This manual gives information about how the software must be installed, which options are available and it gives an in-depth description of every command. This completes requirement [1.9]. The manual can be found in the GitHub repository [11] in the folder *manual*.

[1] Coverage is a metric which is equal to the ratio between the number of lines hit and the total number of code lines (statements) [16].

5

Testing

5.1 Introduction

With the creation of the software, it is possible to easily test the capabilities of the Epson printer to print on CMOS chips iteratively. In this chapter, a run-down will be given of the steps that had to be taken to achieve the remaining requirements specified in chapter 2. These are:

1. The minimum dot diameter should be as small as possible and at most 60 μm.
2. The minimum dot spacing should be as close as possible and at most 141 μm.
3. The accuracy of the dot placement in a grid should be maximised.
4. It needs to be possible to print multiple dots on the exact same position.

All of this needs to be tested on glass slides first and then performed on the CMOS chip surface.

5.2 Evaluating the Dot Size

To make the dot size as small as possible, the parameters in the commands that affect this size should be iteratively modified until the requirement is satisfied. In section 3.4 these commands have been summarised. To check the size of the dots, a PRN file is designed in the software that prints all possible dot sizes for every colour multiple times. This is printed on a glass microscope slide and captured with a digital microscope. The images acquired from this are analysed and important metrics like dot area average, dot area standard deviation, circularity area and circularity standard deviation are calculated. The results from this can be seen in Figure 6.2.

5.3 Evaluating the Dot Spacing

The smallest possible dot spacing should be small to fit as many dots onto the chip surface as possible. This benefits the users by allowing them to measure more points with the chip. To measure the spacing, a grid of dots is printed on a glass microscope slide which is captured by a digital microscope. The spacing of the grid can be specified in the software. The spacing in the raster sent in the `TransferRasterImage` command can only take on 180 DPI (dots per inch) vertically and 180, 360 or 720 horizontally.

This horizontal resolution is plenty as at 720 DPI the spacing from the centre of one dot to the centre of the neighbouring dot is around 35 µm. Using a smaller distance would result in the dots having a too large chance to merge, causing the grid not to consist of single dots, but rather large blobs of combined dots.

The vertical resolution is limited by the geometry of the printhead, which has 1/180 inch spacing between the nozzles in the vertical direction. It is a little on the low side at 180 DPI, as this means a spacing of 141 µm between the dots. To work around this limit, multiple rasters of 180 DPI vertical resolution are printed at small offsets as explained in section 3.4.

While testing, an interesting phenomenon has been observed in the grids: the horizontal spacing tends to be inconsistent. This phenomenon shows itself by grouping two neighbouring dots closer together and then leaving more space to the next pair of dots, which also displays this behaviour. It happens both in rasters of the `TransferRasterImage` command as well as in close spacing of dots specified by the `SetAbsoluteHorizontalPrintPosition` command and is something that is inherent to the Epson ET-8500 printer. This phenomenon can be seen in Figure 5.1.

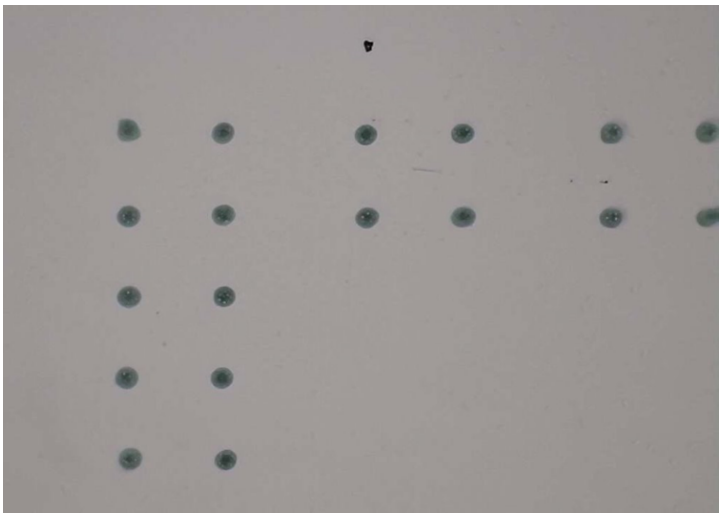


Figure 5.1: Grouping of dots in pairs of two. The space between the first and second columns is smaller than the space between the second and third columns. This keeps repeating in pairs of two.

There are a few ways to work around this which have been used to improve the experiments specified above:

- For horizontal resolutions up to 360, a raster resolution of 720 can be taken leaving every other dot location in the raster set to zero (no dot). This resolves the issue by not printing the dots whose spacing is off.
- If 360 DPI is not enough, splitting the rasters into two rasters of half the horizontal resolution and then applying a small horizontal offset to one of these is an option.

Another, more customizable way of spacing the dots, is by sending only one dot per raster and then directly specifying the position of this dot with the `SetAbsoluteHorizontalPrintPosition` and the `SetRelativeVerticalPrintPosition` command. This method allows for the selection of varying resolutions only limited by the positioning resolution of the printer (which is higher than the maximum resolution in the raster). Unfortunately, this method also results in a horizontal grouping of dots, which can again be worked around by applying small horizontal

offsets to the dots that need it.

5.4 Stacking Droplets

Stacking droplets is useful because it allows the user to increase the thickness of the ink on the CMOS chip and thereby the sensitivity of the sensor. The benefit of stacking small droplets instead of using a single large droplet to form a dot is the fact that the large droplet tends to splash more, causing it to have a larger area with a smaller thickness. Again, having a smaller area per dot is better because it allows the user to fit more dots onto the chip surface. To test stacking the droplets, a 4 by 4 grid is designed in the software where the first dot in the grid consists of 1 droplet and for every subsequent dot, the amount of droplets increments by one such that the final dot consists of 16 droplets. This is first done on a glass microscope slide and can eventually be done on a chip, where one would expect to be able to see the sensitivity increase with the amount of droplets making up the dot.

5.5 Evaluating Mixed Colour Printing

As specified in section 3.4, the colour nozzles aren't perfectly aligned. The use of the shift in the vertical direction to align the colours needs to be evaluated which will be done by printing a grid of dots of varying colour.

6

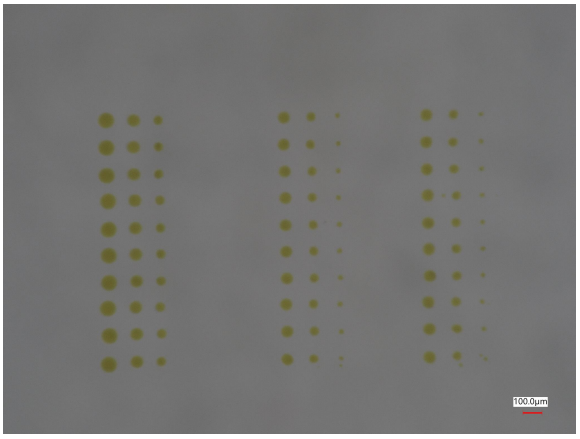
Results

6.1 Introduction

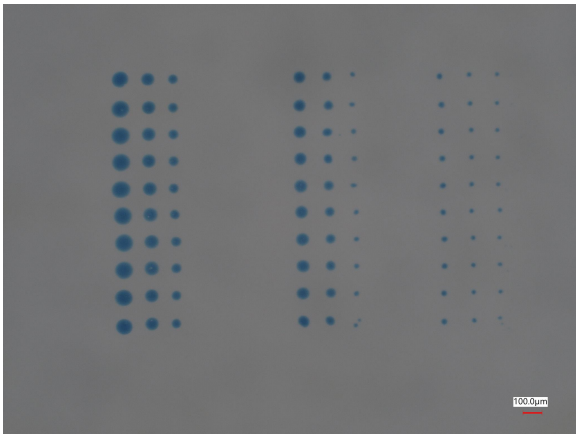
To assess the usefulness of the modified printer for printing on CMOS chips, a series of tests were performed. First, grids of dots were printed onto glass microscope slides. The grids were captured with a digital microscope and analyzed using Python where significant parameters for dot quality and positioning precision were calculated to express the quality of the print. These parameters were calculated and compared for different inks. Also, the spacing between the dots was varied to compare the accuracy at different printing resolutions.
Next, grids of dots were printed onto the CMOS chips directly. These were again captured with a digital microscope.

6.2 Dot Size

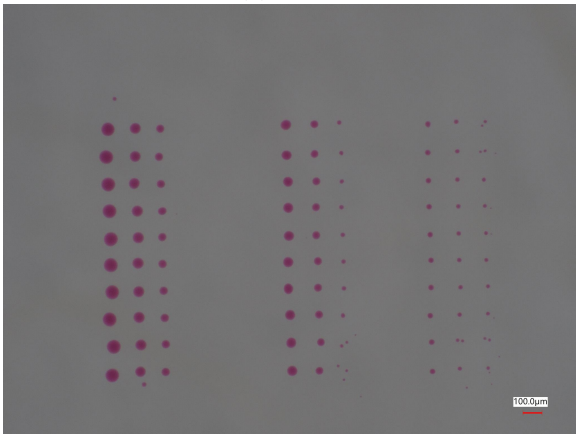
In Figure 6.1, different parameters for the dot size are printed with different colours. The settings go from VSD1 to VSD3 from left to right. For every VSD the first column is large, the second is medium and the last is small.



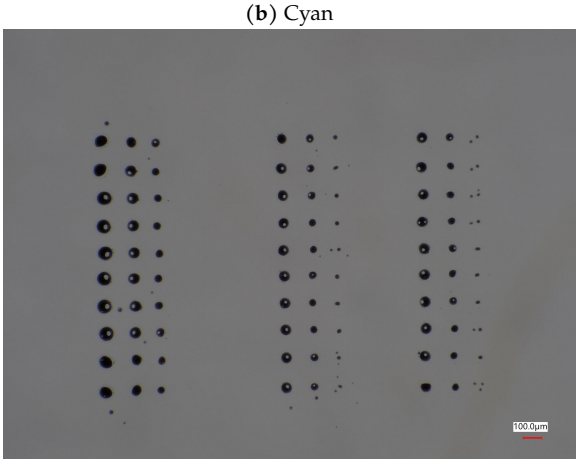
(**a**) Yellow



(**b**) Cyan

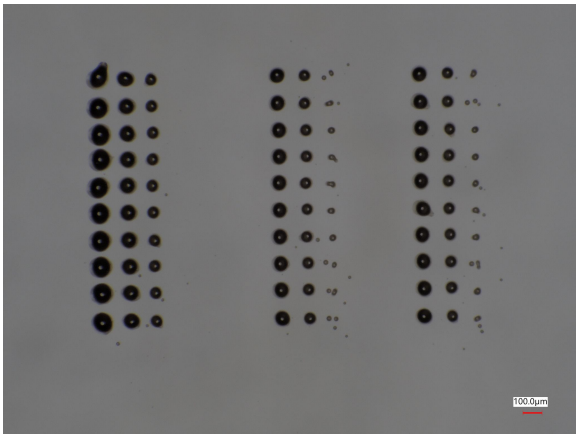


(**c**) Magenta



(**d**) Photo black

(**e**) Grey

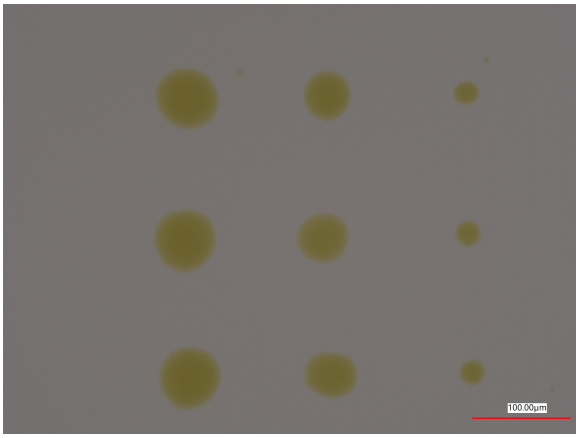


(**f**) Matte black

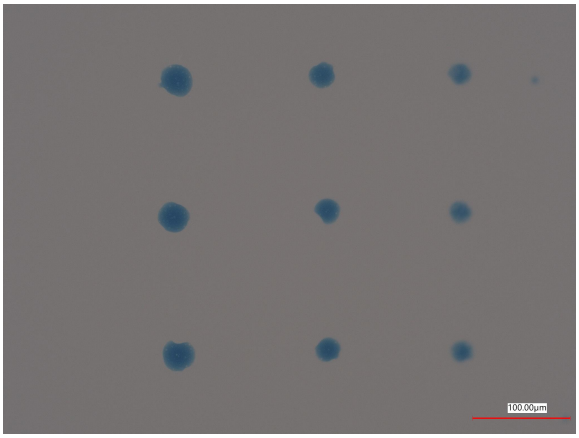
Figure 6.1: VSD comparisons for different colours. From left to right, it goes from VSD1 to VSD3. For every VSD setting a grid of 3x10 is printed with the first column being large, the second medium and the last small.

Visually, the best quality dots were seen when using VSD3. Therefore any further experiments

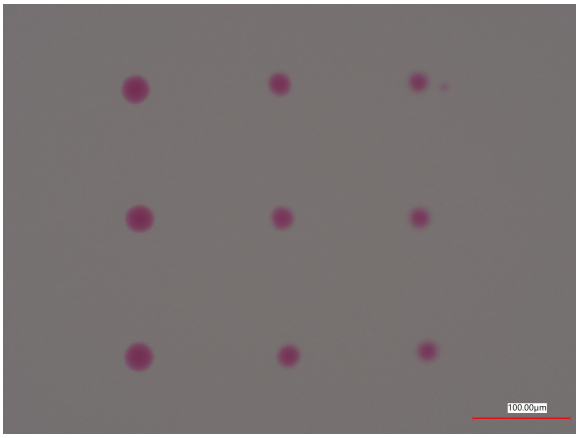
were done using VSD3 dots.



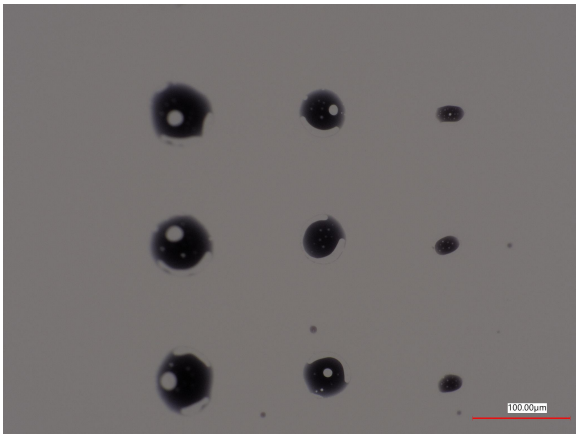
(**a**) Yellow



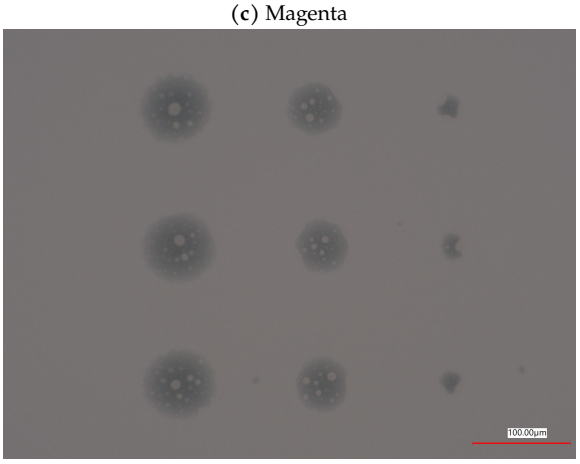
(**b**) Cyan



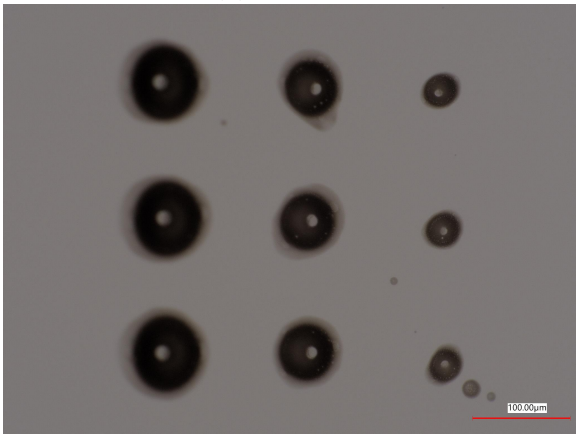
(**c**) Magenta



(**d**) Photo black



(**e**) Grey grid



(**f**) Matte black

Figure 6.2: Zoomed in VSD3 comparisons for different colours. For every colour a grid of 3x10 is printed with the first column being large, the second medium and the last small.

In Figure 6.2, it can clearly be seen that the dots behave differently for different colours. The yellow, blue and magenta dots tend to be the smallest and the others vary more in size. To get the smallest possible resulting dots, all following experiments have been done with the small size in VSD3. Below, in Figure 6.3, a small yellow VSD3 dot can be seen zoomed in.

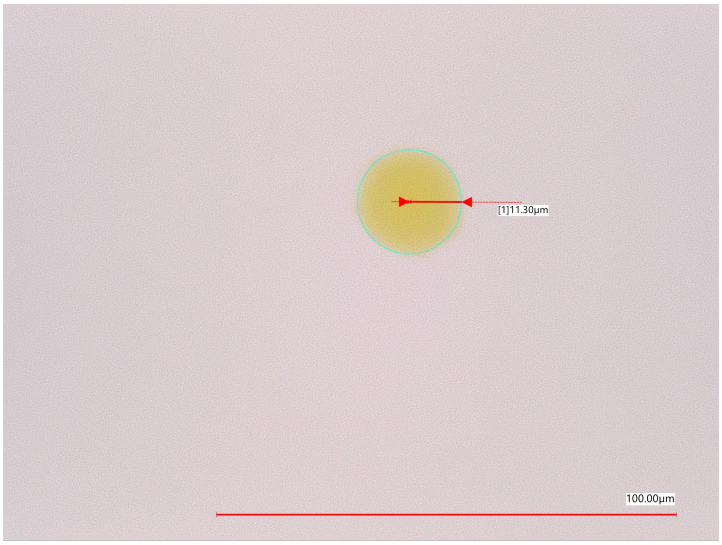
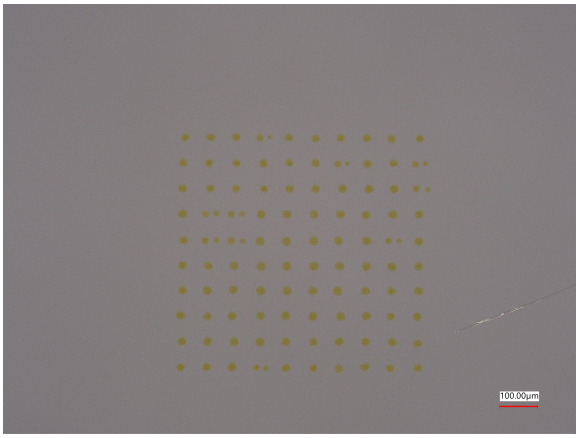


Figure 6.3: Size of a yellow dot with the smallest possible size

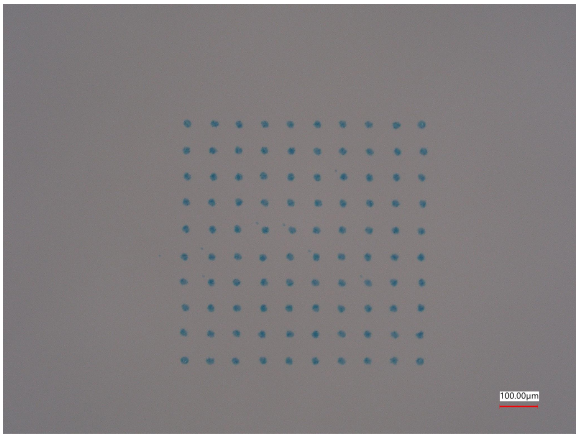
This dot has a radius of 11.30 µm.

6.3 Dot Spacing

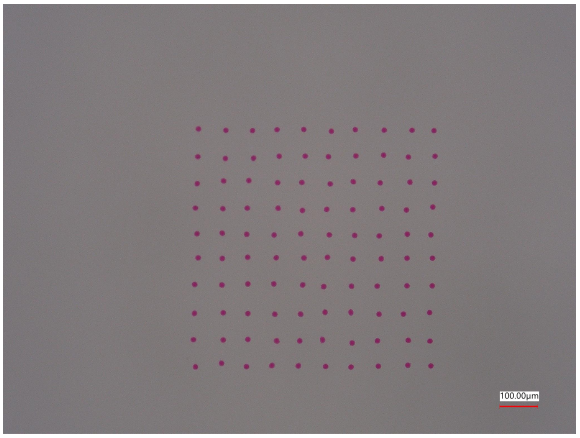
To find the closest spacing possible, rasters of every colour have been printed at different resolutions. The most consistent spacing was at 360 by 360 DPI which means a spacing of about 70 µm both vertically and horizontally. In Figure 6.4 for all different colours a grid of 10x10 is printed with a DPI of 360. The grids have been printed using the method of sending one dot per raster and applying a small horizontal offset to every other dot as described in section 5.3. The reason for choosing this method is that sending complete rasters of dots seems to increase the amount of splitting in every dot which might be due to the higher frequency of releasing the droplets compared to releasing one droplet per raster.



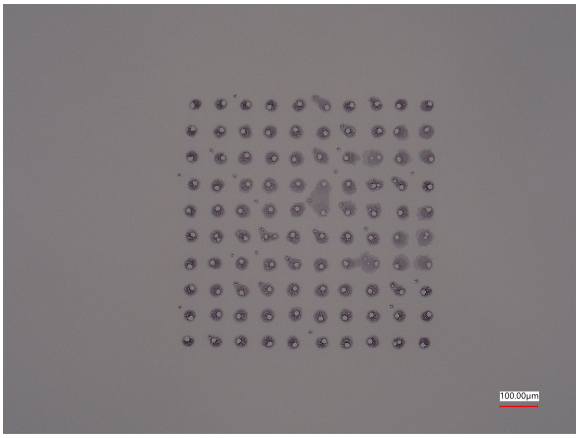
(**a**) Yellow



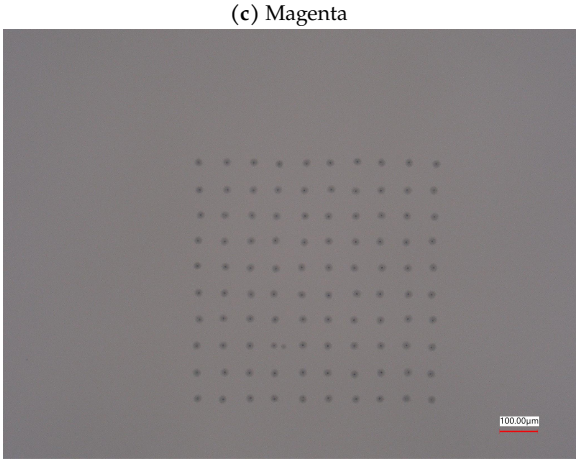
(**b**) Cyan



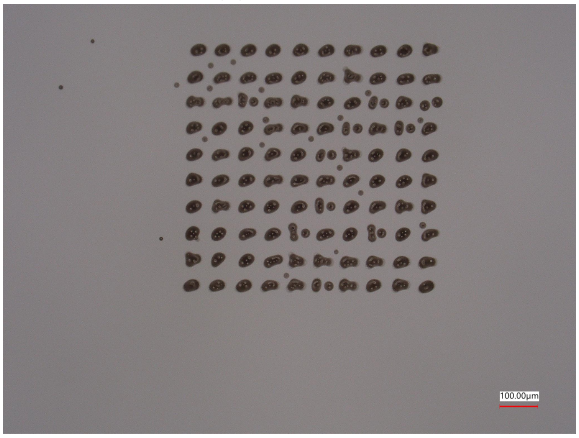
(**c**) Magenta



(**d**) Photo black



(**e**) Grey



(**f**) Matte black

Figure 6.4: 10x10 grid for every colour. The grid has a resolution of 360 by 360 DPI which means a spacing of about 70 µm. This is the closest spacing that we were able to consistently print on.

These grids are analysed and the results can be seen in Table 6.1 and Table 6.2. The measurements are only done for dots that have not spread out or broken up into pieces. The analysis has been done in a custom made Python program seen in Figure 6.5. Here it can be seen that a grid is fitted onto the processed grid of dots and the standard deviation in the vertical and horizontal direction are calculated.

Table 6.1 denotes how accurate the spacing of the dots is and Table 6.2 denotes how consistent

the dots are in diameter and circularity.

Table 6.1: The mean and standard deviation of the horizontal and vertical spacing in the grid for every colour

Colour	Horizontal (µm)	Vertical (µm)
Cyan	69.7 ± 1.9	69.8 ± 1.7
Gray	69.2 ± 3.5	69.4 ± 2.1
Magenta	69.4 ± 4.4	69.1 ± 4.1
Matte black	69.9 ± 2.2	69.9 ± 1.7
Photo black	69.1 ± 2.9	69.7 ± 2.4
Yellow	69.4 ± 1.9	66.8 ± 1.2

Table 6.2: The mean and standard deviation of the diameter and circularity for every dot in the grid for every colour

Colour	Diameter (µm)	Circularity
Cyan	20.6 ± 0.7	0.88 ± 0.014
Gray	21.3 ± 1.8	0.86 ± 0.021
Magenta	14.1 ± 0.4	0.89 ± 0.015
Matte black	45.6 ± 3.6	0.81 ± 0.038
Photo black	39.0 ± 4.4	0.84 ± 0.047
Yellow	23.9 ± 1.0	0.89 ± 0.012

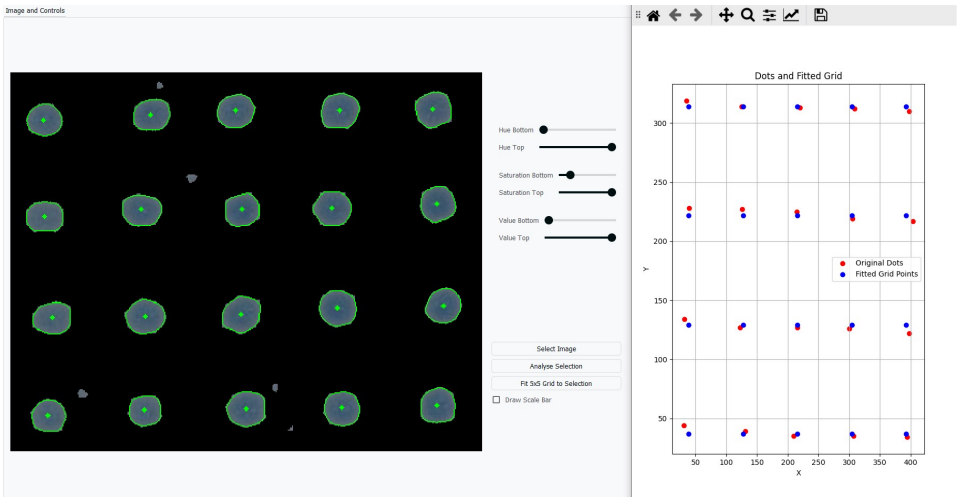
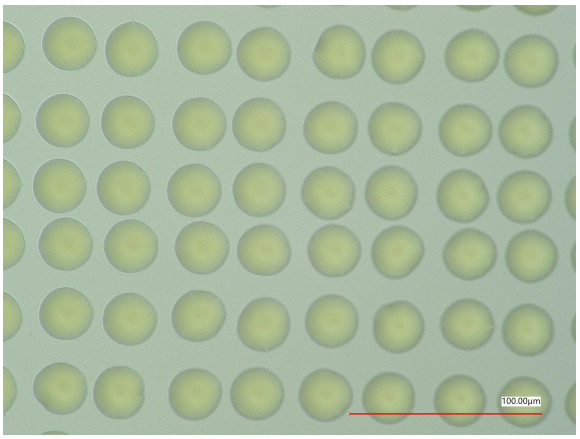


Figure 6.5: Program used to analyze dots of various colours. It allows the user to filter out the dots from an input image taken with a digital microscope.

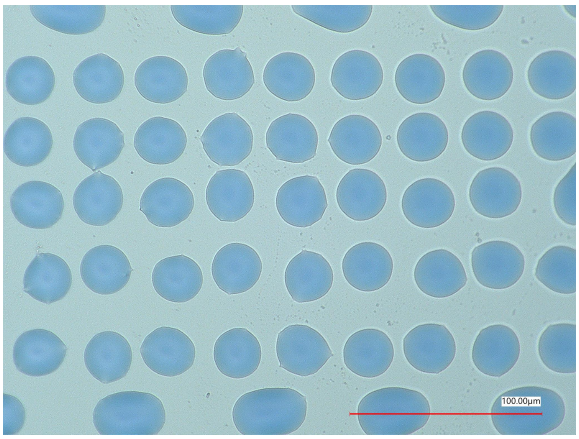
It can be seen that the vertical spacing is more accurate than the horizontal spacing, as expected because the vertical spacing is fixed by the location of the nozzles and the horizontal spacing is determined by the movement of the printhead.

Further attempts have been made at printing higher resolutions (and therefore closer spacing), but these were not as consistent, meaning that a large array had to be printed and good parts had to be taken out of it to prove that it is in fact possible. The reason for this was that the

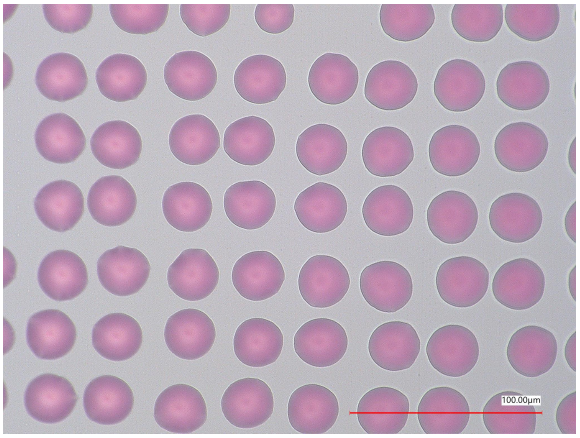
dots were so close to each other that they often recombined. These results are given below in Figure 6.6 for 720 by 720 DPI which means a spacing of about 35 µm both vertically and horizontally.



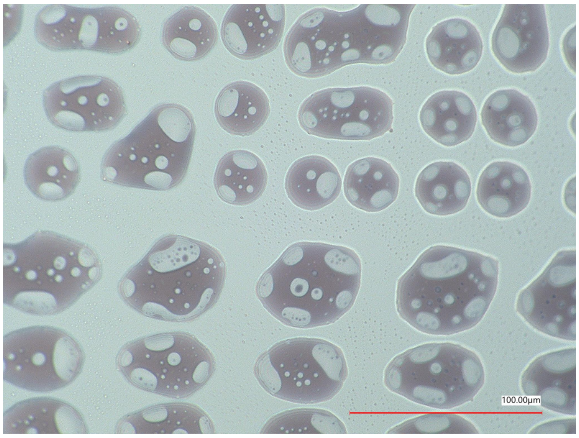
(**a**) Yellow



(**b**) Cyan



(**c**) Magenta

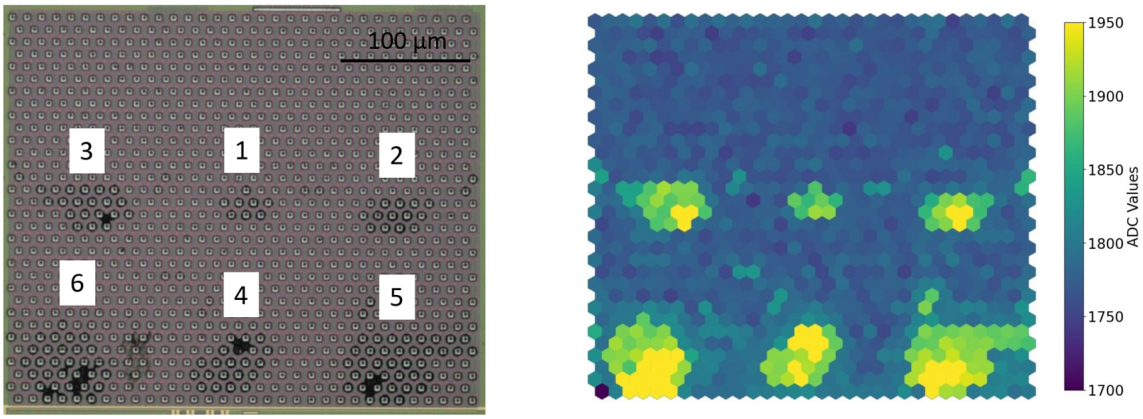


(**d**) Photo black

Figure 6.6: Closest spacing of different colour grids (720 DPI)

6.4 Increasing Number of Dots

To test if the printed dots correctly correspond to the measured signals, a grid is printed where each dot consists of a variable amount of droplets between 1 and 6 (labelled in the picture of the chip). When measuring the chip, it should be visible that the sensitivity increases. Looking at Figure 6.7, it can be seen that the picture on the left, which is a picture of the chip and the picture on the right, which are the measurements of the chip correspond to each other.



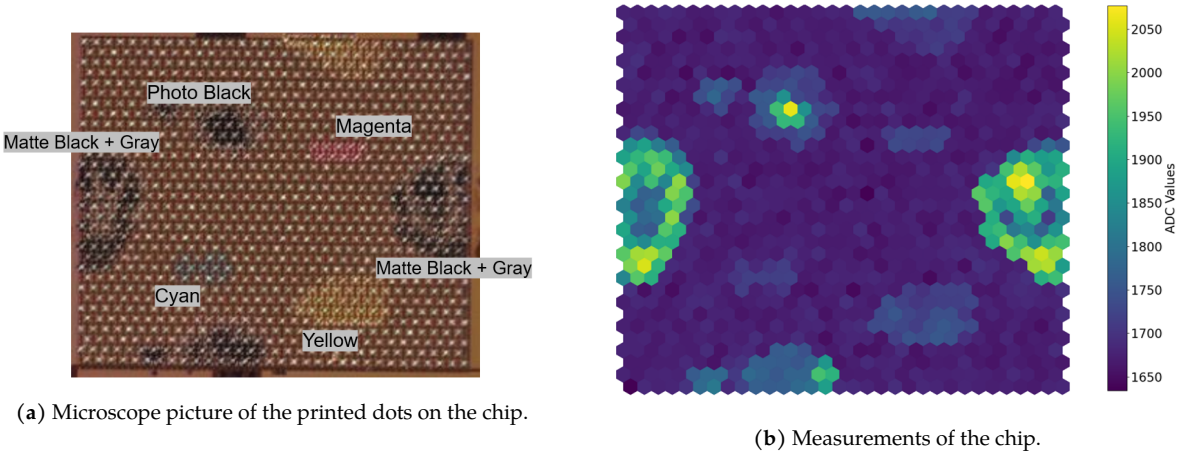
(**a**) Microscope picture of the printed dots on the chip.

(**b**) Measurements of the chip.

Figure 6.7: A chip with a variable amount of droplets printed on labelled with 1 to 6.

6.5 Mixed Colour Printing

To test the printing of different colours together, multiple dots of the small size in VSD3 were placed in a pattern, with each a different colour. This pattern is printed on the chip and can be seen on the left of Figure 6.8.

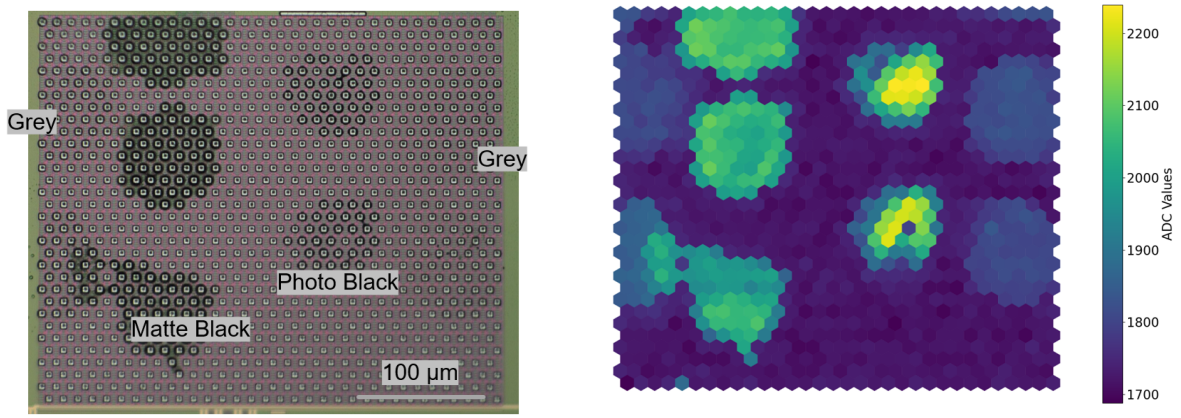


(**a**) Microscope picture of the printed dots on the chip.

(**b**) Measurements of the chip.

Figure 6.8: A chip with all six ink colours of the printer.

As seen in Figure 6.8, the matte black and gray inks merged. This makes it impossible to distinguish between the matte black and gray colours. Therefore another pattern was printed to get good comparisons in the difference of measured signals. The results can be seen in Figure 6.9.



(**a**) Microscope picture of the printed dots on the chip.

(**b**) Measurements of the chip.

Figure 6.9: A grid of multiple ink colours.

It is hard to see the gray colour on the chip, but looking at the measured signals it can clearly be seen that there exists a fluid and that all colours behave differently.

7

Conclusion

7.1 Conclusions

The goal of the software subgroup was to develop a user-friendly interface that allows users to print individual dots with multiple inks using the Epson ET-8500.

To see if the results are successful the requirements as stated in section 2.2 are checked one by one. The first three requirements consist of the software being a graphical user interface and when connected to the printer via a USB cable can print the patterns [1.1] [1.2] [1.3]. These requirements are all succeeded and without these, the end result would be useless. Requirement [1.4] states that the software must be installable on Windows or Linux. This is both possible and can be done using instructions from the software manual. Requirements [1.5] [1.6] [1.7] [1.8] all go about being able to print a grid with different colours on the CD tray, they are all satisfied and more in-depth instructions can be found in the software manual, which was also a requirement [1.9].

The other requirements are about the limitations and capabilities of the ET-8500 using the software. Using the results from chapter 6, the following capabilities are achieved. The smallest dot size possible is around 22.60 μm in diameter, with a spacing of 720 DPI (35.3 μm), but this spacing cannot be printed consistently due to recombination of dots. Also the properties of the droplet vary clearly from ink to ink. The closest consistent spacing is 360 DPI which means a spacing of (70.6 μm). This is well within the requirements. Furthermore printing dots on top of each other has been proved useful as a clear relation was seen between the sensitivity of the CMOS chip and the amount of dots printed on a single spot. The accuracy of the dot placement is good but has a lot of potential to improve. Especially the horizontal placement has shown large inconsistencies. Printing multiple inks on the chip was successful but needs finetuning as the raster was not exactly dispersed as expected.

7.2 Future Work

The resulting print qualities look promising. With the current state of the software, it is possible to print all the patterns one could imagine. However, they are not all as easy as they could be to create. To improve this the following work can still be done:

- Make it possible to draw dots on a raster instead of having to specify with a matrix of ones and zeros where dots need to go.

- Add the functionality to select a dot, modify it, and see the properties.
- Print using grids of single dots to make it possible to print in any DPI which is not restricted to the raster resolutions available by the printer.
- Do more research about the falling of the droplets which influence the quality of the printed dots.
- Test with other inks that have a smaller spread, which makes it more appropriate for printing dots close to each other.
- Test with other Epson printers to see if it also works for other models.
- Create a calibration step for new printers. Using the already-made software for analysing the dots, the offsets of the positioning can be used to create a calibration profile for the printer which would adjust these offsets in the printing process.

References

[1] M. A. Shah, D.-G. Lee, B.-Y. Lee, and S. Hur, "Classifications and applications of inkjet printing technology: A review," *IEEE Access*, vol. 9, pp. 140 079–140 102, 2021. DOI: 10.1109/ACCESS.2021.3119219. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2021.3119219 (visited on 04/23/2024).

[2] *Ecotank et-8500 datasheet / brochure*, Seiko Epson, Jun. 2024.

[3] R. Waasdorp, O. van den Heuvel, F. Versluis, B. Hajee, and M. K. Ghatkesar, "Accessing individual 75-micron diameter nozzles of a desktop inkjet printer to dispense picoliter droplets on demand," *RSC Adv.*, vol. 8, pp. 14 765–14 774, 27 2018. DOI: 10.1039/C8RA00756J. [Online]. Available: http://dx.doi.org/10.1039/C8RA00756J (visited on 04/23/2024).

[4] R. Scafè, P. Auer, P. Bennati, *et al.*, "Production of radioactive phantoms using a standard inkjet printer and the public domain multi-printing code genia," *Physica Medica*, vol. 27, no. 4, pp. 209–223, 2011, ISSN: 1120-1797. DOI: https://doi.org/10.1016/j.ejmp.2010.10.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1120179710000621.

[5] R. Scafe, P. Bennati, P. Auer, *et al.*, "Multi-printed inkjet phantoms for radionuclide molecular imaging," in *2008 IEEE Nuclear Science Symposium Conference Record*, 2008, pp. 1081–1087. DOI: 10.1109/NSSMIC.2008.4774587.

[6] T. Shimoda, K. Morii, S. Seki, and H. Kiguchi, "Inkjet printing of led polymer displays," *MRS Bulletin*, vol. 28, Nov. 2003. DOI: 10.1557/mrs2003.231.

[7] *Epson programming guide for 4 color epson ink jet printer*, Aug. 2016.

[8] mrnuke, *Epson inkjet driver esc/p-r*, https://github.com/mrnuke/epson-printer-escpr-improved, 2017.

[9] *The developer's guide to gutenprint*, 2003. [Online]. Available: https://gimp-print.sourceforge.io/reference-html/book1.html.

[10] Apple, *Cups*, https://www.cups.org/, 2007–2021.

[11] Thomas-164, *Escp2-client*, https://github.com/Thomas-164/ESCP2-Client, 2024.

[12] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697.

[13] T. Q. Company, *Pyqt5 reference guide*, https://www.riverbankcomputing.com/static/Docs/PyQt5/, 2023.

[14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of resuable object-oriented software*. Addison-Wesley, 1995.

[15] github, *Github*, 2024. [Online]. Available: https://github.com/.

[16] V. Khorikov, *Unit Testing: Principles, Practices, and Patterns*. Shelter Island, NY: Manning Publications, 2020, ISBN: 978-1617296277. [Online]. Available: https://www.manning.com/books/unit-testing.

A

Chip Dimensions

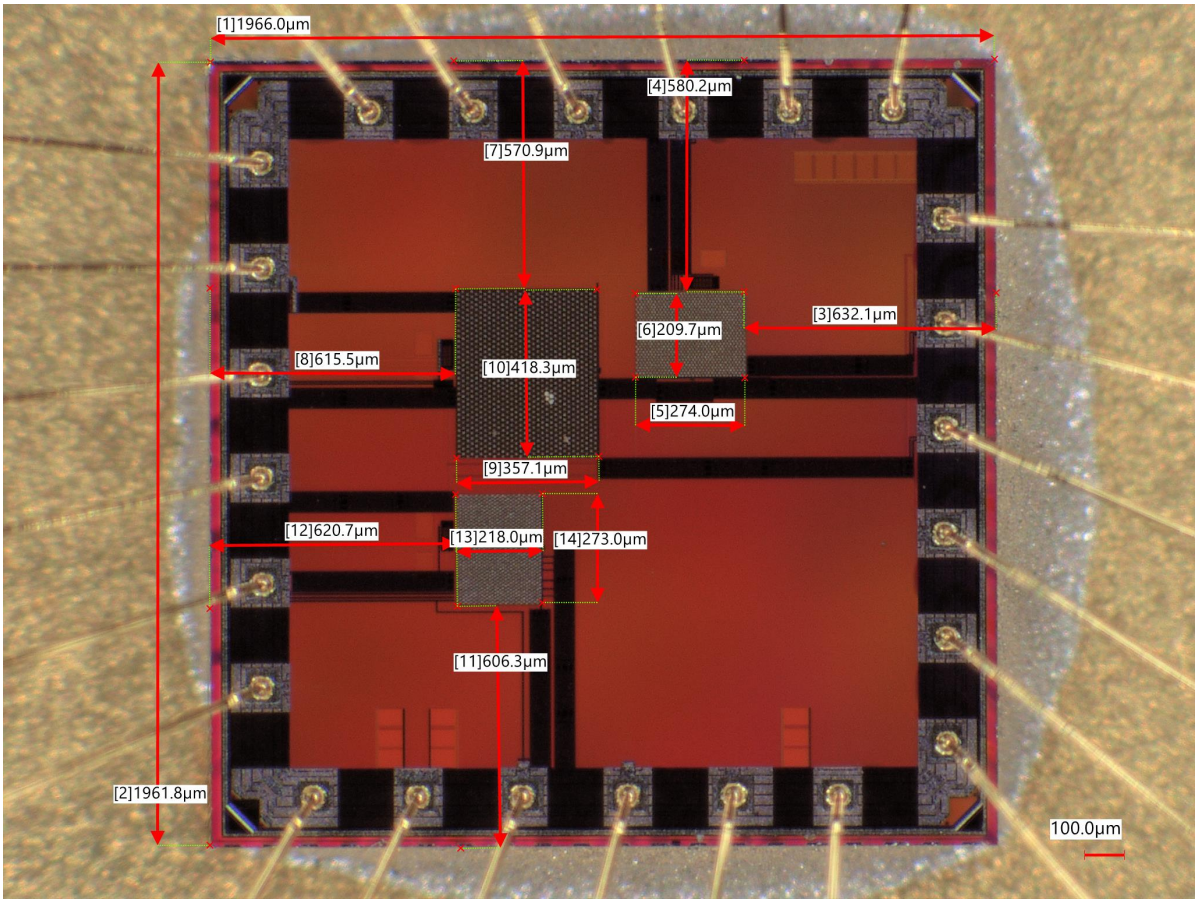


Figure A.1: The dimensions of the chip printed on. Three sensor arrays can be seen with sizes of 418.3 µm by 357.1 µm, 274.0 µm by 209.7 µm and 218.0 µm by 273.0 µm.