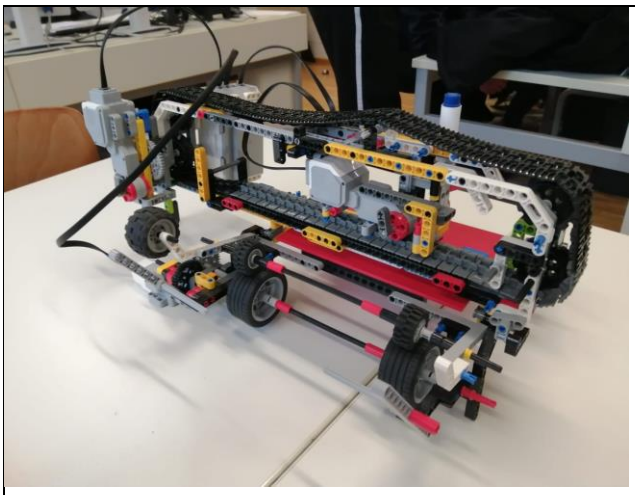
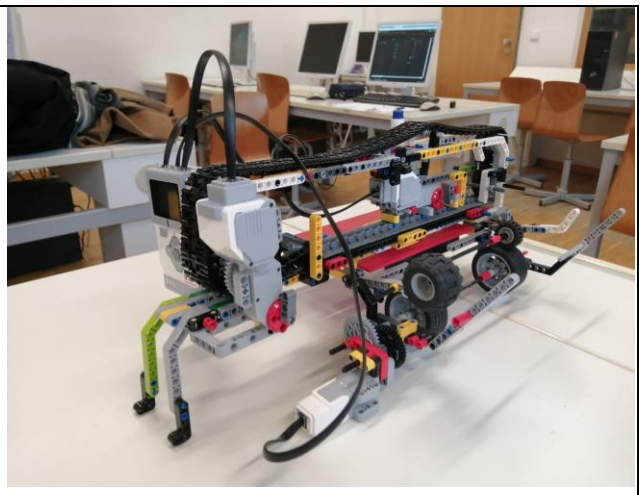
Criterion B: Design

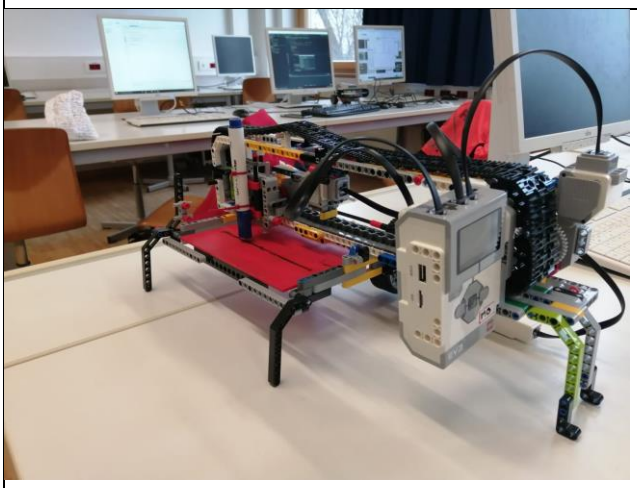
System I – Printer

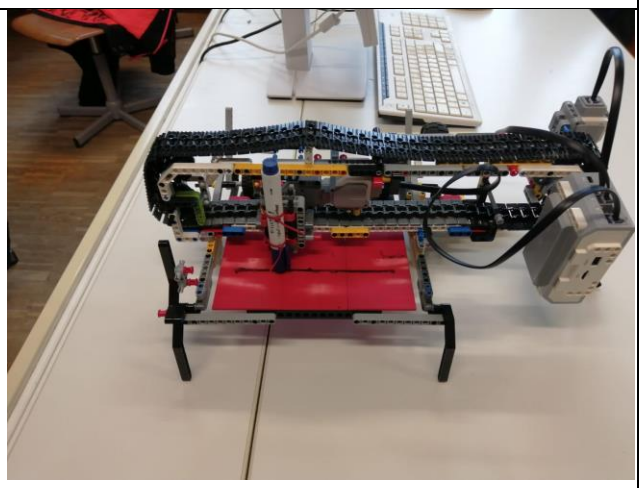
Prototype / Mock-up

I created a prototype first as a proof of concept. It was a simplified version of my final product, as it had lower printing resolution. It was able to print low-resolution images (Max 100×100px) at a decent speed (~5-10min per print).









Design Plan

The final printer would consist of 3 different parts.

Part-A consists of a paper holder and a set of rails. The rails will allow “Part-B” to move along them, giving the printer movement along the y-axis.

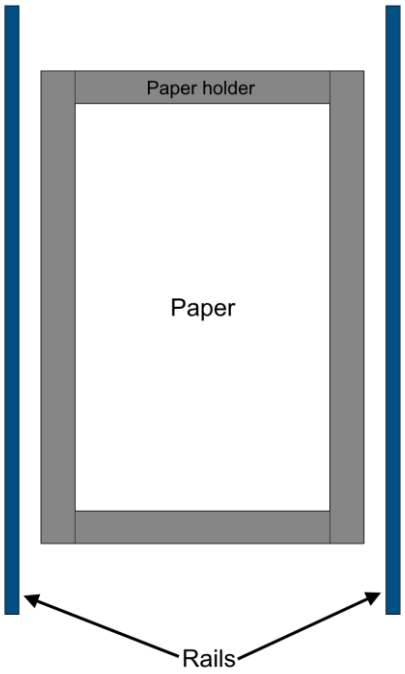


Diagram: Part-A

Part-B includes 4 wheels, two connected to an axle, connected to a motor. This allows “Part-B” to move along the rails of “Part-A”. “Part-B” also has a set of rails where “Part-C” will move along.

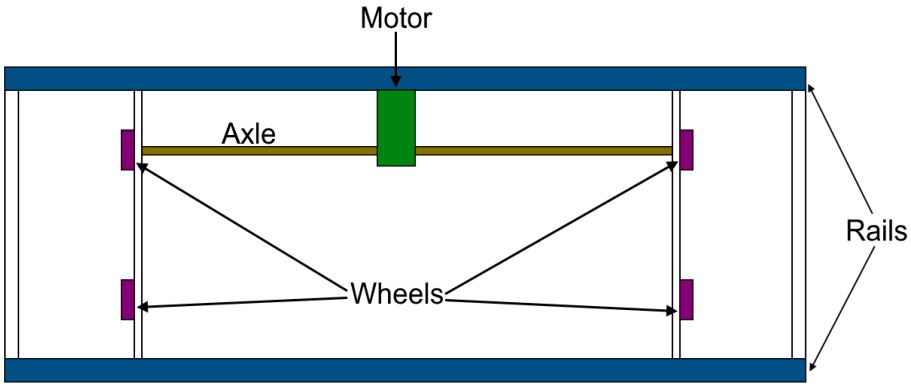


Diagram: Part-B

Part-C includes 4 wheels, with 2 connected to an axle, connected to a motor. Allowing movement along the rails of “Part-B”, giving the printer movement along the x-axis. The “Mindstorm Brick” is the main processor of the printer, the motors are connected to it and the printing program is executed by it.

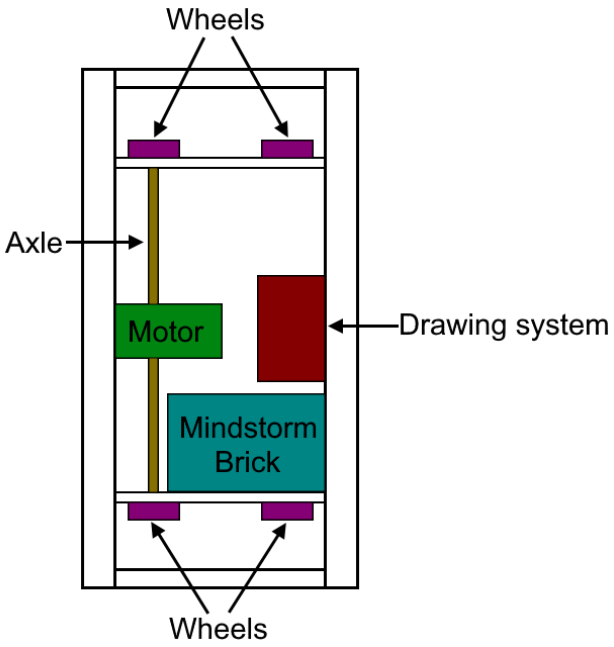


Diagram: Part-C

Printer movement

Y-axis movement

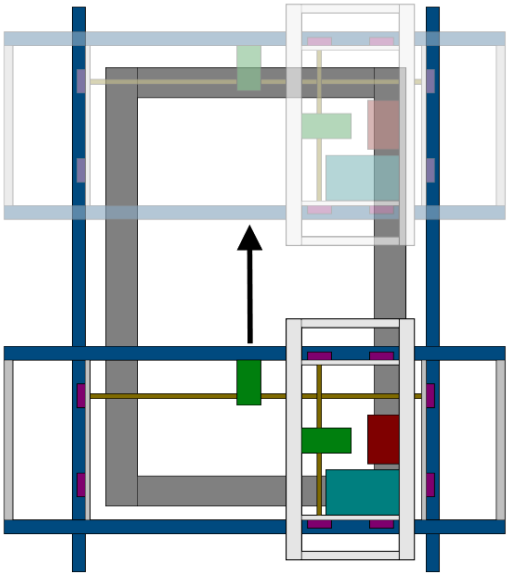


Diagram: Part-B moving along Part-A's rails

X-axis movement

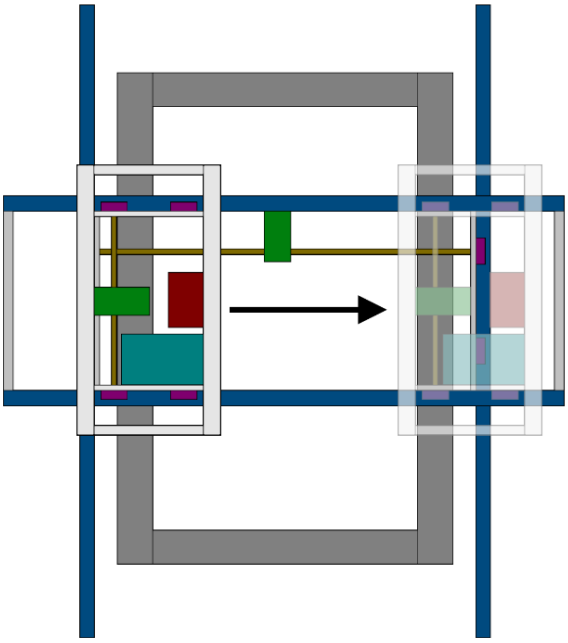


Diagram: Part-C moving along Part-B's rails

Realized Parts

Part-A

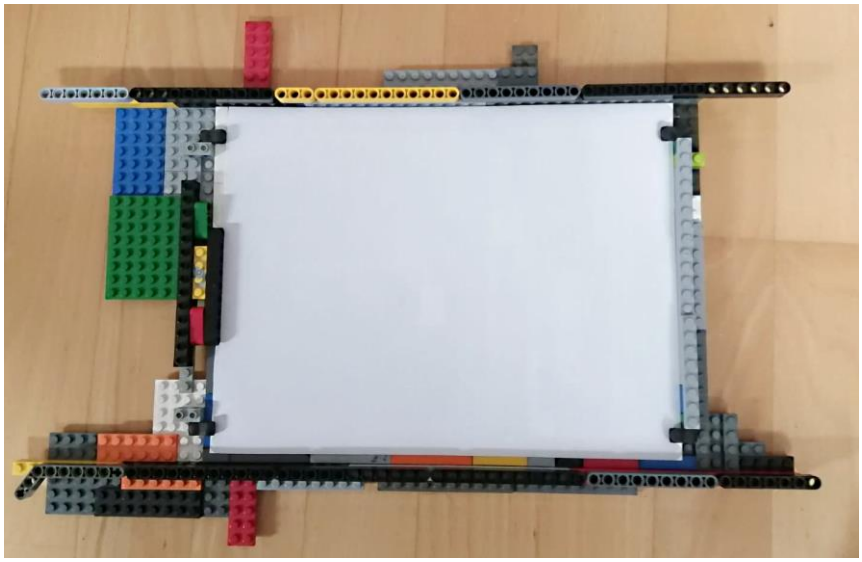


Image: Top-down view

Part-B

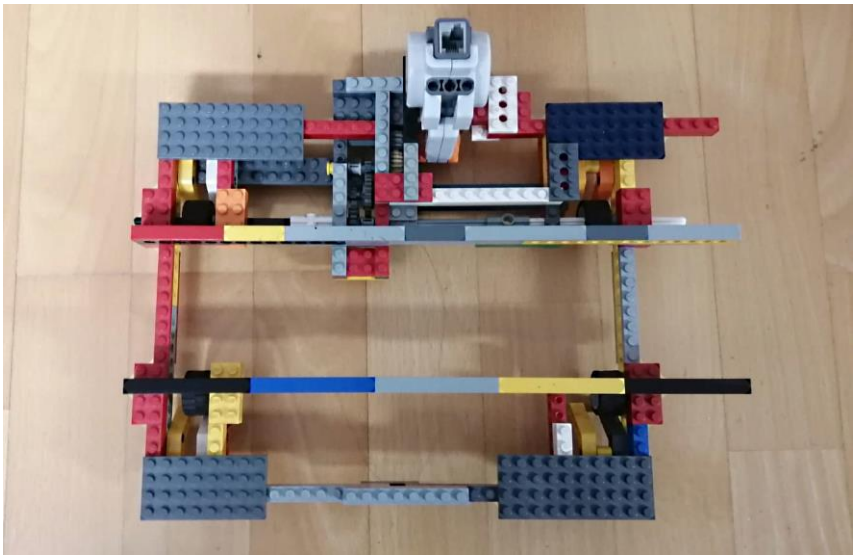


Image: Top-down view

Part-C

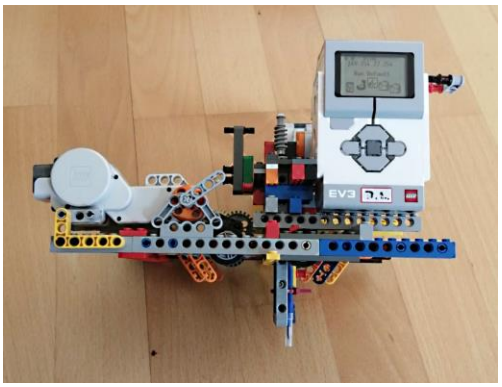


Image: Side view with "Brick"

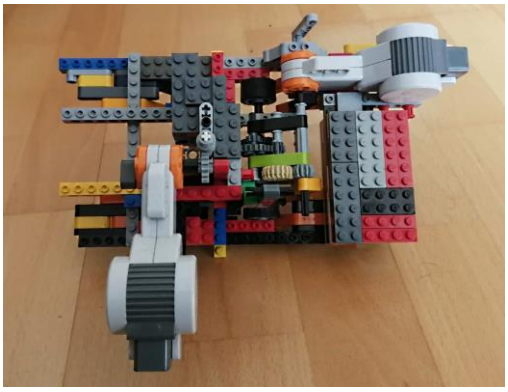


Image: Top-down view without "Brick"

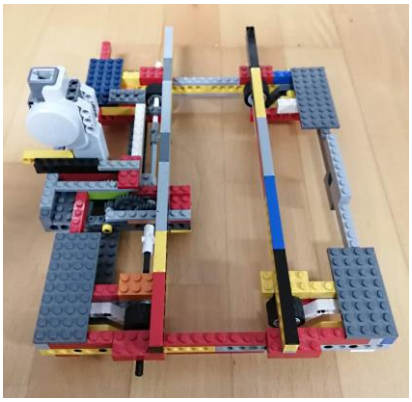


Image: Side view

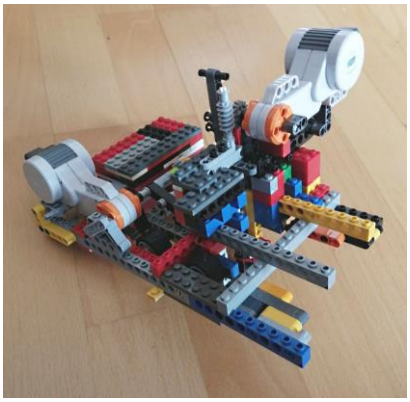


Image: Side view without "Brick"

Parts together

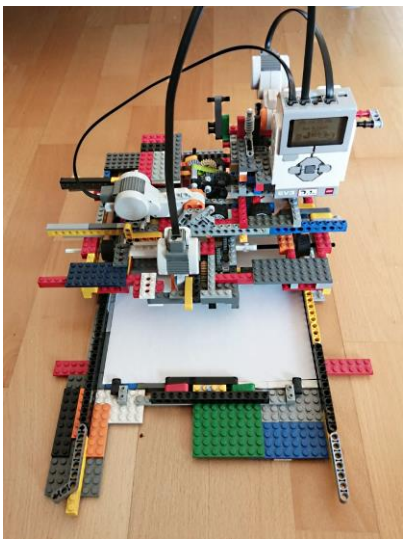


Image: Front view

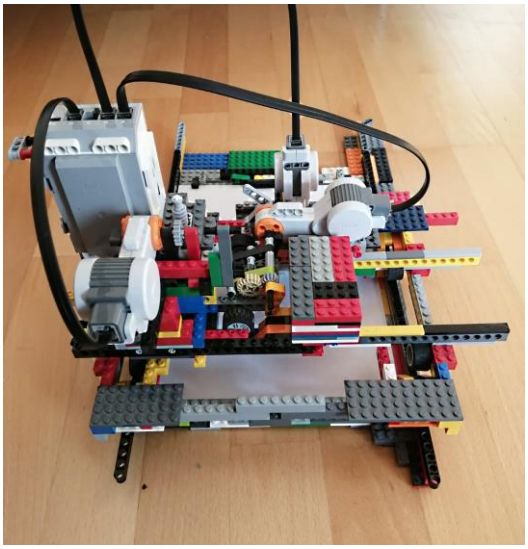


Image: Back view

Drawing system

The printer works by drawing dots at certain positions, to be able to do so I needed a system that allowed me to move a pen upwards and downwards.

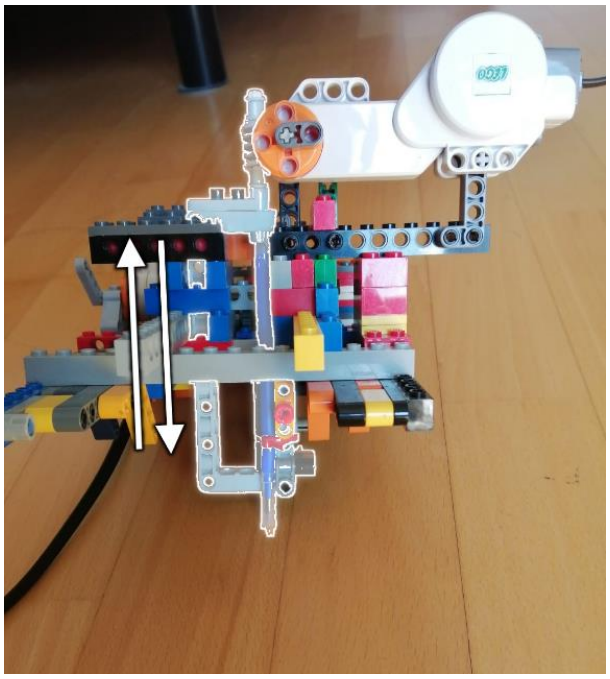


Diagram: Highlighted part can move vertically, independently of "Part-C"

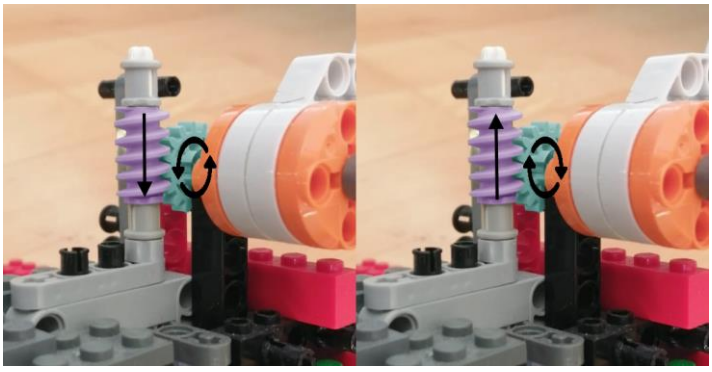
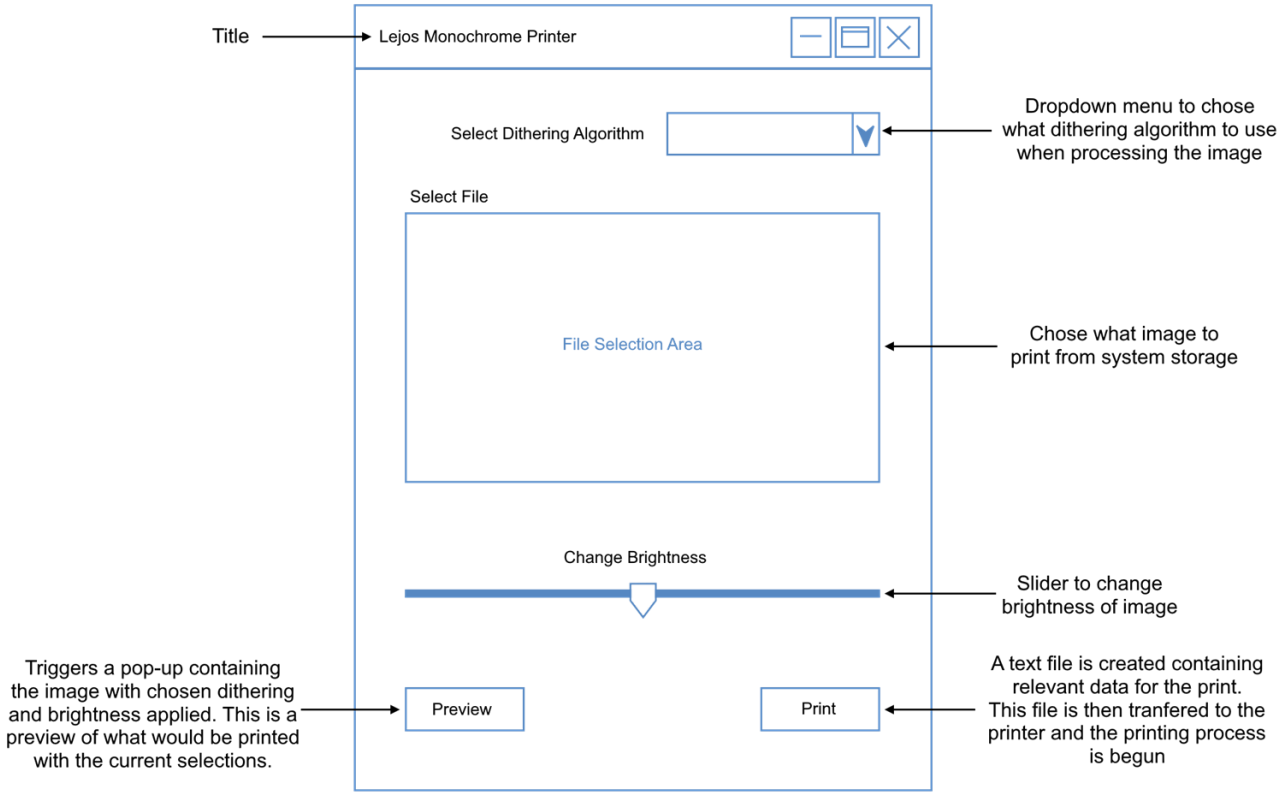


Diagram: Gear setup allowing both upwards and downward motion

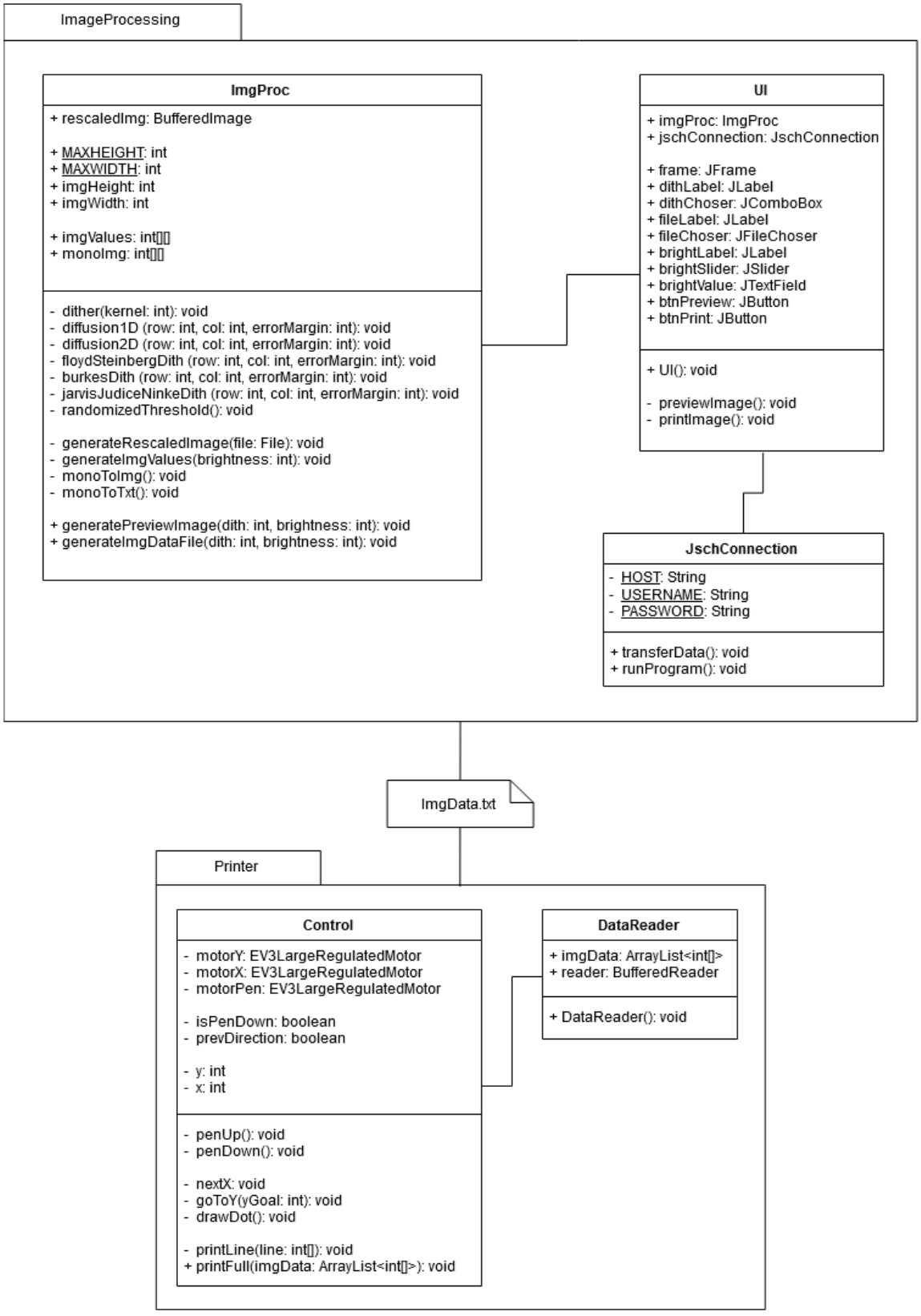
System II – Image Processing

This system is run from an external system (PC, laptop etc.) and allows users to choose what image they intend to print. There are a few additional settings that affect how the image is converted into black and white (dithering algorithms & brightness). All these settings are accessible from a UI.

User interface mock-up



UML Diagram


ImageProcessing
ImgProc
+ rescaledImg: BufferedImage
+ MAXHEIGHT: int
+ MAXWIDTH: int
+ imgHeight: int
+ imgWidth: int
+ imgValues: int[][]
+ monoImg: int[][]
- dither(kernel: int): void
- diffusion1D (row: int, col: int, errorMargin: int): void
- diffusion2D (row: int, col: int, errorMargin: int): void
- floydSteinbergDith (row: int, col: int, errorMargin: int): void
- burkesDith (row: int, col: int, errorMargin: int): void
- jarvisJudiceNinkeDith (row: int, col: int, errorMargin: int): void
- randomizedThreshold(): void
- generateRescaledImage(file: File): void
- generateImgValues(brightness: int): void
- monoToImg(): void
- monoToTxt(): void
+ generatePreviewImage(dith: int, brightness: int): void
+ generateImgDataFile(dith: int, brightness: int): void
UI
+ imgProc: ImgProc
+ jschConnection: JschConnection
+ frame: JFrame
+ dithLabel: JLabel
+ dithChoser: JComboBox
+ fileLabel: JLabel
+ fileChoser: JFileChoser
+ brightLabel: JLabel
+ brightSlider: JSlider
+ brightValue: JTextField
+ btnPreview: JButton
+ btnPrint: JButton
+ UI(): void
- previewImage(): void
- printImage(): void
JschConnection
- HOST: String
- USERNAME: String
- PASSWORD: String
+ transferData(): void
+ runProgram(): void
ImgData.txt
Printer
Control
- motorY: EV3LargeRegulatedMotor
- motorX: EV3LargeRegulatedMotor
- motorPen: EV3LargeRegulatedMotor
- isPenDown: boolean
- prevDirection: boolean
- y: int
- x: int
- penUp(): void
- penDown(): void
- nextX: void
- goToY(yGoal: int): void
- drawDot(): void
- printLine(line: int[]): void
+ printFull(imgData: ArrayList<int[]>): void
DataReader
+ imgData: ArrayList<int[]>
+ reader: BufferedReader
+ DataReader(): void

Class functionality

ImageProcessing

UI

User interface for the program. Can choose image to print, select dithering algorithm and brightness.

ImgProc

Chosen image is reformatted to be usable by the printer. Includes making image monochrome, resizing, applying dithering algorithms and brightness. The image data is then saved onto the text file "imgData.txt".

JschConnection

Transfers the "imageData.txt" file to the printer using a ssh connection. Also starts the printing process by running the jar file on the printer.

Printer

Data Reader

Reads "imageData.txt" and creates an ArrayList of integer arrays with the image data.

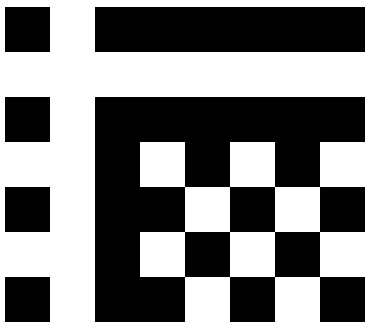
Control

Prints the image using the previously generated ArrayList. Contains all the methods responsible for controlling the motors of the printer.

ImgData.txt

This file contains the data that the printer requires to print the image. The text file consists of 0's (white pixels) and 1's (black pixels).

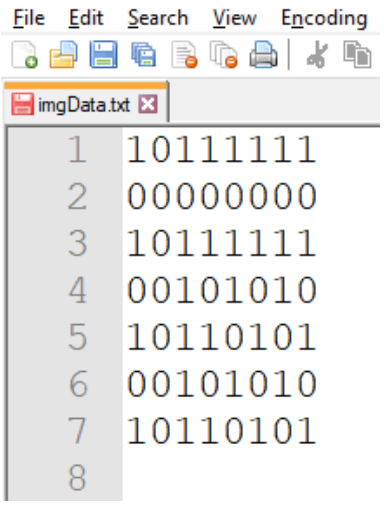
Example:



Example image

1	0	1	1	1	1	1	1
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	0
1	0	1	1	0	1	0	1
0	0	1	0	1	0	1	0
1	0	1	1	0	1	0	1

Image values



imgData.txt file containing values

Process description