University of Southern Denmark

Group 3

Start Date: 1. November 2022
Due Date: 14. December 2022

Instructor: Lars Carøe Sørensen, lcs@mmmi.sdu.dk

Submitted By:

Alan Rashid
alras22@student.sdu.dk

Thomas Vilholm
thvil22@student.sdu.dk

Anas Butt Hussain
anhus22@student.sdu.dk

Jakub Hubert Rudowski
jarud22@student.sdu.dk

1 Introduction

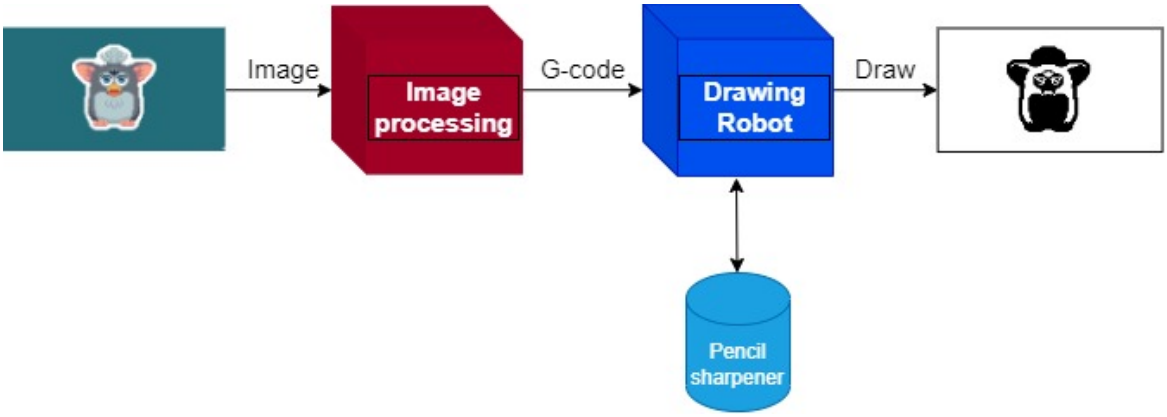

Figure 1: Flow chart of the process

The flow chart shows the process of converting a digital image into a drawing. It will be the thread of which the report follows, where the individual parts will be explained further in later sections below. Simply put, an image is chosen which is imported to a program; This is where the image processing happens. Once the image processing is finished a g-code file is then extracted and uploaded to the program of the drawing robot. It is important that the file uploaded is a g-code file. The reason for this is that the program of the robot only understands g-code. It takes the g-code as input for its movement pattern to make the robot able to draw. In addition to this, there is also the fact that the pencil starts to blunt. Therefore, sharpening is also needed for a nice and clear image. The pencil sharpener is activated through a sensor and is in result contactless.

A few experiments will be done to test different parameters of the robot in terms of drawing quality and time efficiency. The drawing method chosen for this project is the so called "dot-method", which is how the robot draws, and under which the experiments are carried out.

Contents

1 Introduction **1**

2 Project description **3**

2.1 Problem analysis: . . . 3

2.2 Problem formulation: . . . 3

2.3 Problem limitations: . . . 3

3 Image processing **4**

3.1 Image perception . . . 4

3.2 Shortest Tour . . . 4

3.3 G-Code Layout and variation in pencil force . . . 6

3.4 Partial discussion . . . 8

3.5 Partial conclusion . . . 8

4 Pencil sharpener **9**

4.1 Voltage regulator . . . 9

4.2 Pencil sharpener circuit . . . 11

4.3 Partial discussion . . . 13

4.4 Partial conclusion . . . 14

5 Data processing **15**

5.1 Experiment with drawing speeds . . . 15

5.2 Experiment with picture types . . . 17

5.3 Experiment with picture resolutions . . . 21

5.4 Partial discussion . . . 22

5.5 Partial conclusion . . . 23

6 Discussion **24**

7 Conclusion **25**

8 Appendix **26**

9 Bibliography **27**

2 Project description

The following three subsections act as a way for the reader to figure out the intention and thought-process behind the project, after reading this, it should be clear what the intentions, limitations and ideas for the report are.

2.1 Problem analysis:

The intention of the project is to make a program in which an imported image gets drawn by a drawing robot. In regards to this there will in the following report strictly be used "Dot-method". There will be an analysis of which drawing speed delivers the highest quality picture. Different kinds of pictures such as: landscape photos, character art, and more will be used and compared to determine what the robot can draw best. There will be an analysis of the inner workings of the program and why the chosen functions are there. There will also be a section dedicated to the circuit of the electronic pencil sharpener.

2.2 Problem formulation:

In the report the following points will be scrutinized

- How can the program convert a digital image into a physical drawing, by using the drawing robot?
- How can the program scale to a fitting size so that the robot does not go past its borders?
- How to increase the effectivity of the "Dot-method" program?
- As the pencil is drawing it also needs to be sharpened. How does the robot determine when to sharpen?
- How is it possible to convert 12 Volts dc to 6 Volts dc?
- The robot itself does not have a sensor for knowing when to sharpen. It is therefore necessary to incorporate an external electronic solution.

2.3 Problem limitations:

The project has the following limitations:

- Mathematica has been chosen instead of MATLAB for the image processing.
- The size of the platform that the robot draws on, restricts the size of the image.
- For the pencil sharpener a simple contact solution is not allowed.

3 Image processing

An image is imported from the internet and scaled to fit a realistic size for the robots drawing platform, while not sacrificing too much detail, then it gets analyzed by the Mathematica inbuilt functions to be exported in a G-Code format, shown in (fig. 2). The individual parts of the flow chart are explained systematically. The Mathematica code is found in the appendix.

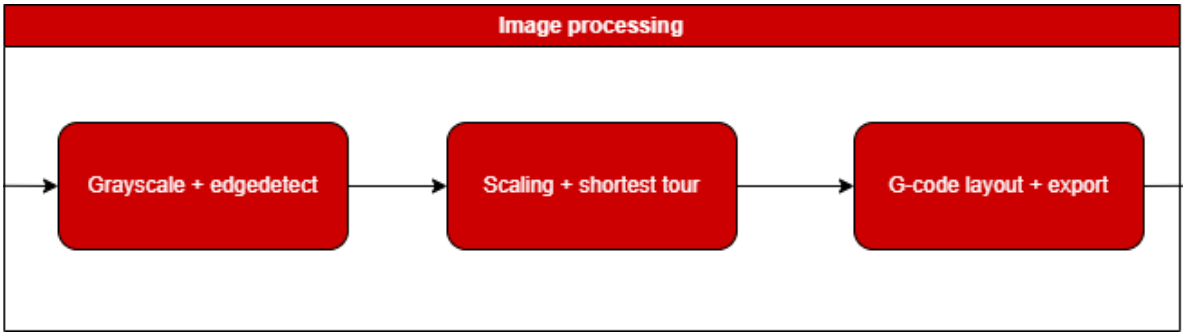

Figure 2: Flow chart of the image processing steps

3.1 Image perception

The imported image is analyzed by the Mathematica code. As shown in the code, the imported image gets color converted into gray scale, by the inbuilt Mathematica function ColorConvert. The ColorConvert function uses 2 parameters, the image and the color scale. Edges from the color converted image are determined by the inbuilt Mathematica function EdgeDetect. The edgedetected version of the image is then used to draw by the robot, that means the drawing will only be drawn by the edges of the color converted image. Edge detect takes two parameters, the image and the scale in form of a pixel range. The scale in the EdgeDetect function was set to different values for every image type, see section 5. The scale was set to a value, which presented the image edges most visible. EdgeDetect makes the whole process more efficient, because the function determines how many pixels are extracted from the image.

3.2 Shortest Tour

The inbuilt Mathematica function PixelValuePositions, extracts all of the pixels, from the edge detected image, as X and Y coordinates. The function uses two parameters, an image and a color. Edges after the EdgeDetect function are in white color, which in return makes white the second parameter in PixelValuePositions. X and Y coordinates are extracted from the PixelsValuePositions function so they can be applied to scaling.

The scaling function is applied by scaling the coordinates instead of the picture. The coordinates are scaled so that the minimum value is (0,0) and maximum is (100,100). These scaled coordinates are deducted by 50 so that the minimum and maximum value of coordinates are (-50,-50) and (50,50), (see CoordinatesLayer1 in the code). The reason for these values is that the robot range was set to a square, with its corners in (50,50), (-50,50), (-50,-50) and (50,-50), (see (fig. 3) for a better understanding of the robots drawing dimensions). The square dimension made it easier for the robot to scale the picture. The DeleteDuplicates and Round Mathematica inbuilt functions, make it so that the coordinates are rounded to integers and so that there are no duplicates of the same coordinates. These inbuilt functions remove the excess points on the drawing, but keep the shape of the image visible.

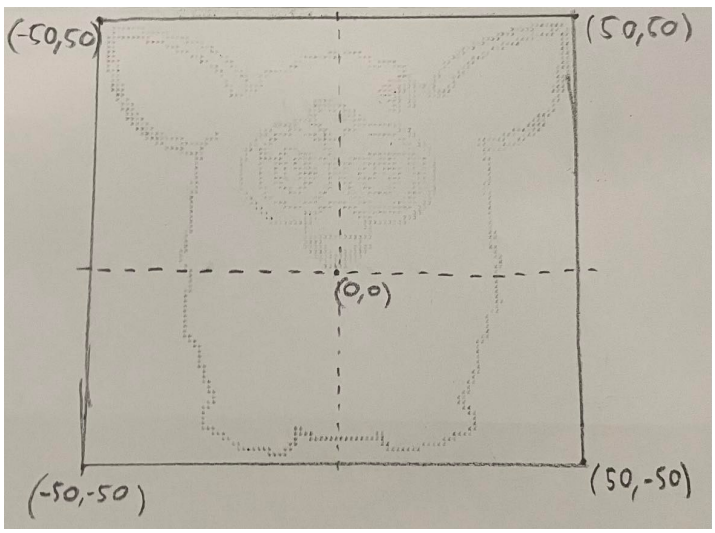

Figure 3: Dimensions of the robot drawing space

FindShortestTour is an inbuilt Mathematica function and it makes the drawing process by the robot quicker and thus more effective. The input for the function is coordinates, it then orders the coordinates by the closest distance between each coordinate, so that the robot can efficiently draw the picture. The same picture was drawn with and without the FindShortestTour function. The picture without the function was drawn in one hour and thirty-five minutes, the picture with the function was drawn in 40 minutes, see (fig. 4 and fig. 5, the second parameter of EdgeDetect was set to a value of 2.5.)

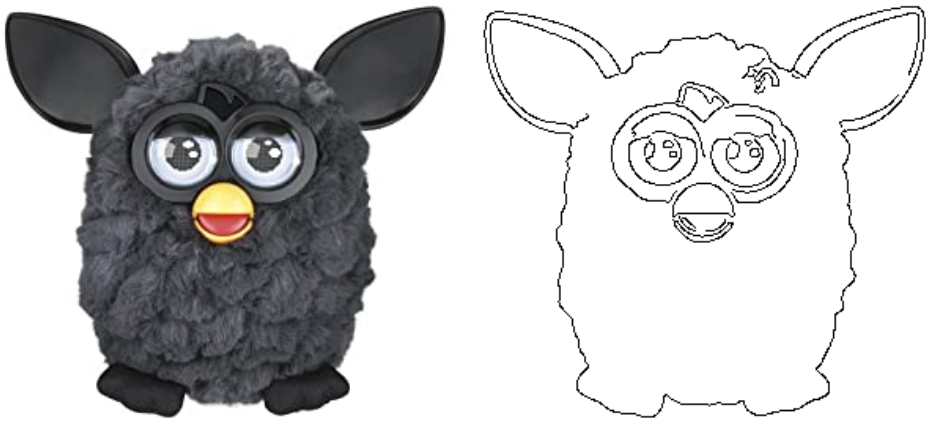

Figure 4: Picture of Furby, from left to right: Imported image, Edge detect of the imported image

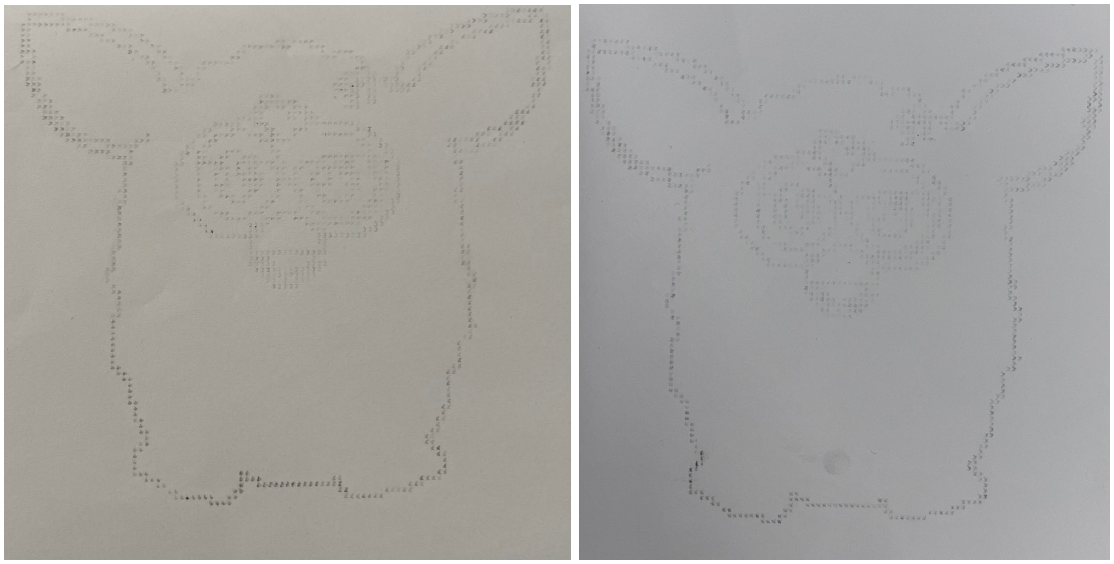

Figure 5: Picture of Furby, from left to right: picture without the ShortestTour function, picture with the ShortestTour function.

3.3 G-Code Layout and variation in pencil force

The layout of the G-code was made by combining strings or variables converted into strings by the ToString inbuilt Mathematica function. The way "dot-method" works is that for every scaled coordinate, the robot draws a dot on the drawing. G01 g-code was when the dot was drawn and G00 g-code was when the pencil was in the air. The height was set to 20 when drawn in G01 and 32 when floating in the air in G00, all in Z values, so that the pencil would move to the next dot in the air. Sharpening a pencil made the pencil a bit shorter, and that is why there had to be a deduction of Z values after sharpening. The sharpening was activated when G98 was implemented. Variation

in pencil height, after using the pencil sharpener, is of most importance.
The height of the pencil determines how much pressure is put on a piece of paper and how visible the lines are. There was an experiment conducted, which determined that values of Z should be deducted by 0.2 every time it had sharpened the pencil. The observations were documented, see (fig. 6). The program is set to sharpen the pencil every 800 points, so that the points are visible.

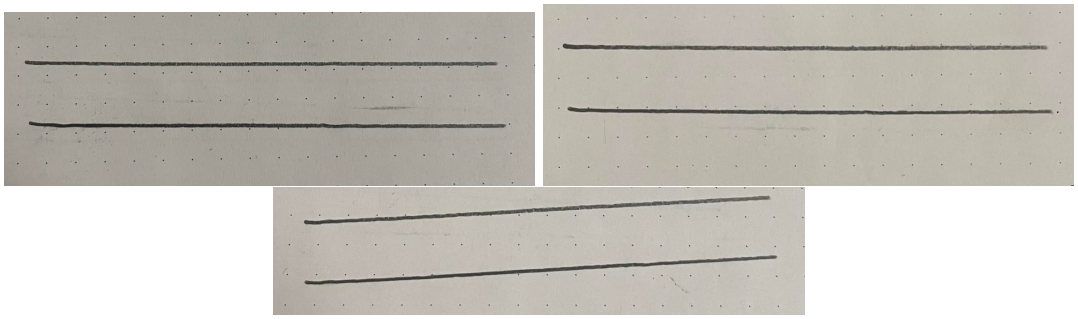

Figure 6: Experiment: Varying height of pencil after sharpening, from left to right: 0.2, 0.5, 1 deducted from Z after sharpening

A few other inbuilt functions were used to set the layout of the g-code in a table. The Riffle inbuilt Mathematica function used on tables will cause the two tables to mix with each other, this function is used four times in the code. The function takes a second parameter, which is a number, and it mixes the two tables with the second parameter value. In this way G00, G01 and G98 could be connected in one table, so it could be extracted in a .txt file. The extracted file from the Mathematica code was uploaded to the robot interface, and then the robot would start drawing. After 800 dots made, the robot sharpens the pencil by executing G98 g-code. Section 4 describes how the sharpening of the pencil was made possible.

3.4 Partial discussion

The variation in pencil height was solely decided by the Z coordinates instead of real life measurements. The usage of real life measurement could potentially make it more accurate, which was not tested. The main problem lies in the dimensions of the square, which was set from -50 to 50 in (x,y) coordinates. The robots working radius was 100 mm, but the robot interface would not allow the g-code to exceed the range of -50 to 50 in (x,y). The size could arguably be determined by the robots max radius limit. The experiment with FindShortestTour, see (fig. 5), emphasizes the problem with efficiency. The robot is much quicker when FindShortestTour is in use. While drawing, the problem with variation in pencil height appeared, after sharpening the pencil. It was solved by an experiment, see (fig. 6), where pencil height was deducted by 0.2 in Z values after sharpening; found by comparing the results.

3.5 Partial conclusion

Concluding the image conversion problems brought up initially The digital image is converted into the physical drawing by using the Mathematica code. There are a lot of inbuilt functions in the Mathematica code, mentioned in section 3.1, section 3.2 and section 3.3. These functions analyze the imported image and generate a code from the pixel coordinates, which are scaled to fit the drawing size of the robot. The drawing limit of the robot was set from -50 to 50 in (x,y) coordinates. The picture had to be scaled down to fit the square, see (fig. 3). The robot was programmed to sharpen the pencil every 800 points it draws. The program also sets the height of the pencil down so that the dots continue unaltered in contrast. The effectivity of the method used was mostly based on FindShortestTour. But also the varying second input of EdgeDetect, which determined edges of the image. This input was based on visibility of the original picture. This would determine how many pixels were extracted from the image.

4 Pencil sharpener

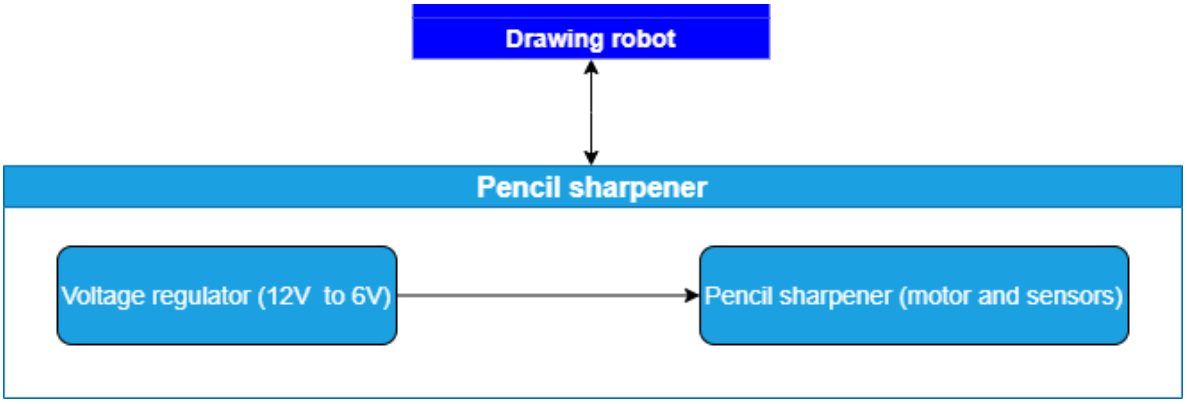

Figure 7: Flow chart of the pencil sharpener

The circuits were divided into two. The reason for this was to simplify the process of building the circuits. Because of the separation it also allowed for a focus on the function of the specific circuit. The voltage regulator was designed to decrease the voltage from 12 to 6 without having to keep the motor and sensing in mind. The pencil sharpener circuit could be created without having to worry about the voltage input, the goal of this circuit was to have 6 Volts across the dc motor only when the pencil is in the hole of the sharpener.

4.1 Voltage regulator

Voltage regulation is necessary for delivering the correct voltage to the pencil sharpener, the output from the drawing robot is 12 Volts and 1 Ampere, and the motor running the sharpener only needs 6 Volts and 1 Ampere. The transistor "LM7805" is a fixed-voltage regulator designed for a wide range of applications. This component is essential in the circuit [9.1]. The two diodes used in the circuit not only force the direction of the current, but also function as small resistors.

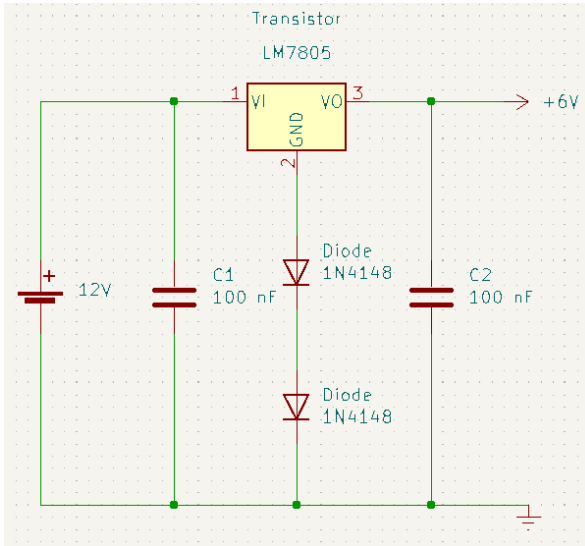

Figure 8: KiCad drawing

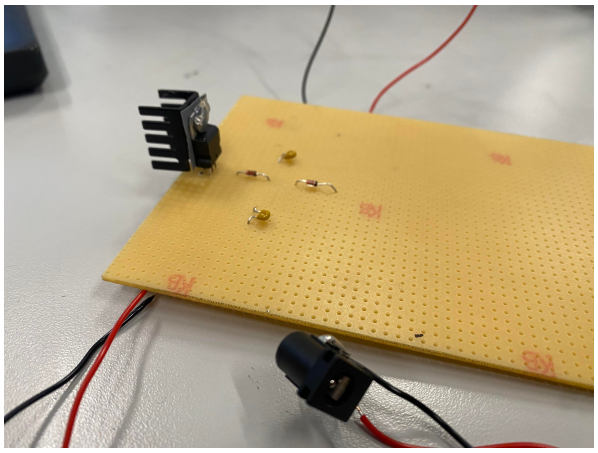

Figure 9: In real life picture

List of components used for voltage regulation:

- LM7805 transistor (see appendix item 5)
- 2x 1N4148 diodes (see appendix item 2)
- 2x 100nF capacitors
- 12V DC power connector jack

4.2 Pencil sharpener circuit

The setup is a redesign of an online circuit[9.2] and is made so that the voltage is inversely proportionate to the distance of the obstacle from the infrared diodes. As the pencil approaches the infrared diodes, more of the infrared light gets reflected back to the photo which results in more current flowing. The potentiometer has been used to set a reference point. If current is equal to or above the reference then current goes from the output of the operational amplifier into the ground pin of the transistor, allowing current to flow through the motor.

The operational amplifier "LM2904" works as a comparator comparing the voltage across the 10 kΩ resistor to the voltage reference point set by the potentiometer, whenever the voltage at non-inverting input is more than the voltage at the inverting input, the output will turn on. When no object is near the diode pair, the motor needs to be off. So the potentiometer is adjusted so as to make the voltage at inverting input more than non-inverting.

The transistor "BC639 NPN Transistor" is current-controlled and acts as a switch, "turning on" when it receives the input from the operational amplifier into its ground pin allowing current to flow through its other terminals.

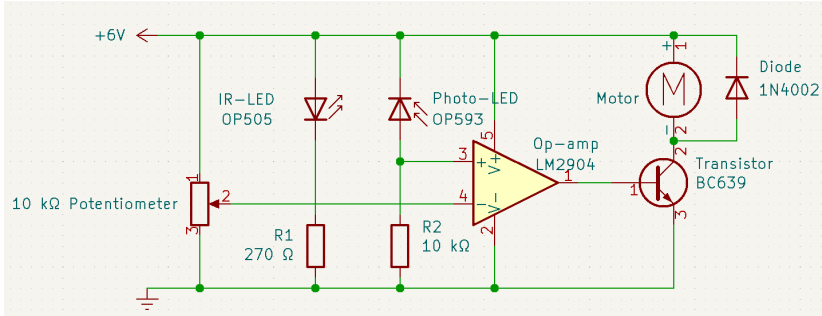

Figure 10: KiCad drawing

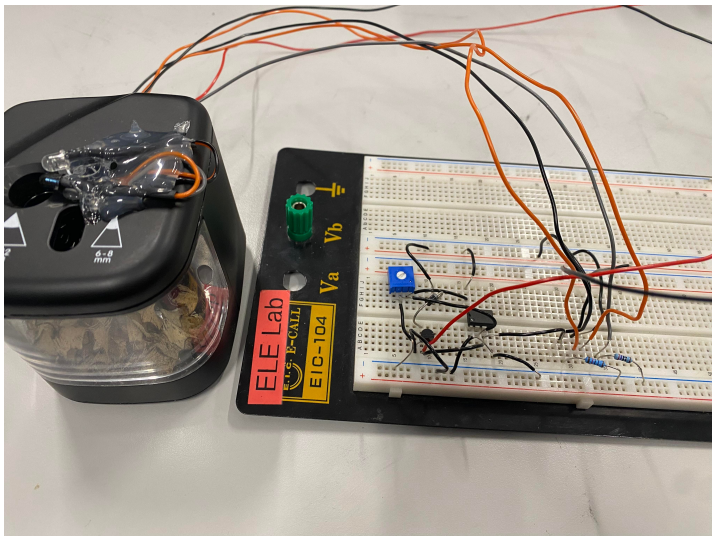

Figure 11: In real life picture

Components used for the pencil sharpener circuit:

- 270 Ω Resistor
- 10 kΩ Resistor
- InfraRed Diode PhotoDiode pair, OP505A IR-LED and OP593 Photo-LED (see appendix item 6 and item 7)
- LM2904 Operational amplifier (see appendix item 4)
- BC639 NPN Transistor (see appendix item 3)
- 10 kΩ Potentiometer
- 6V DC Motor
- 1N4002 Diode (see appendix item 1)

4.3 Partial discussion

The positioning of the infrared diode pair proved to be a major challenge, because the receiver can be very sensitive to infrared light; it needs a minimal amount of infrared light to turn on. Of course the sensitivity can be adjusted by increasing the amount of resistance in the potentiometer, but if the resistance is too high the infrared light reflected off of the pencil wont be enough to make the operational amplifier cause an output. The diode pair is therefore glued carefully in place next to each other besides the pencil sharpener hole, so that when the pencil descends the infrared light from the emitter accurately hits the pencil and reflects back to the receiver.

Another challenge discovered through usage was that the transistor used in the voltage regulation "LM7805" needed cooling. Numerous times the transistor overheated and the pencil sharpener would get 12 Volts unregulated. The thermal problem was fixed with a cooling solution strapped to the transistor.

It was discovered that the motor will not accept the transistor as a switch if it does not have a diode in parallel. This is because the magnetic field created by the motor will not have a way of dispersing its energy, additionally if the diode is reversed the motor will not work.

It could be discussed whether a potentiometer was necessary or not. Essentially the potentiometer is a resistor of a varying size. It was used to set a reference point, which once crossed, turned on the motor. On first glance it could seem like a very useful component. However once the optimal resistance is found it is essentially obsolete. It proved to be quite troublesome, because during testing it would often change the reference point resulting in the pencil sharpener not operating. This would also make it hard to pinpoint what the problem was.

4.4 Partial conclusion

No switch is allowed so the solution incorporated is a operational amplifier that only has an output if the voltage across the receiver is at the reference-voltage created by the potentiometer. When the transistor then receives the output from the operational amplifier through its ground pin, current is allowed to flow to the motor. The voltage comes from the infrared emitter and receiver. The voltage which goes to the operational amplifier is directly proportional to the amount of infrared radiation falling on the receiver. The closer an object is the more gets reflected back. Glue was found to be the superior solution for holding the 2 infrared diodes in place on top of the pencil sharpener.

The conversion from 12 Volts to 6 Volts was made possible with the transistor "LM7805", which provided the necessary voltage. The transistor converts the 12 Volts from the input into 6 Volts to output. This was necessary because the pencil sharpener cannot handle more than 6 Volts.

5 Data processing

This section is dedicated to the experiments of the drawing robot. Different parameters have been changed and tested to see if there were any notable differences in quality. Three experiments were carried out with each experiment having around 3 to 4 images drawn.

5.1 Experiment with drawing speeds

For the first experiment the variable of speed was changed. The first attempt was at the standard 100 percent speed, the second attempt was at 200 percent speed and lastly at 400 percent speed. For the following experiment a picture of a furby was used. It starts off by going through edgedetect with a second parameter of 2.5.

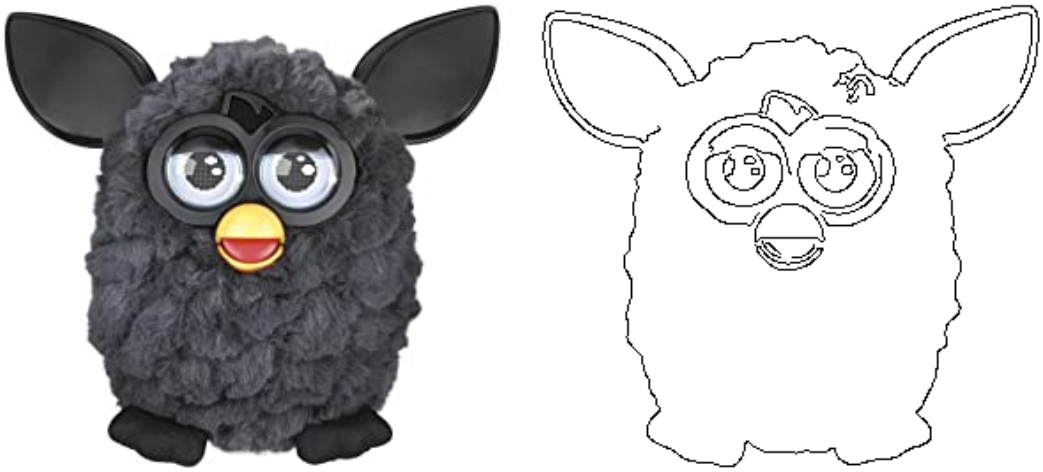

Figure 12: From left to right: Picture of Furby, Edge detect of Furby

Figure 13: From left to right: 100%(duration: 40 minutes), 200%(duration: 20 minutes), 400%(duration: 10 minutes) robot speed

5.2 Experiment with picture types

For the second experiment 4 different images were selected. The four images were respectively a Furby with EdgeDetect value of 2.5, Popeye with EdgeDetect value of 4, Mona Lisa with EdgeDetect value of 10 and Mount Everest with EdgeDetect value of 11.5.

Figure 14: Picture type Furby, from left to right: Imported picture of Furby, Edge detect of Furby, Drawing of Furby

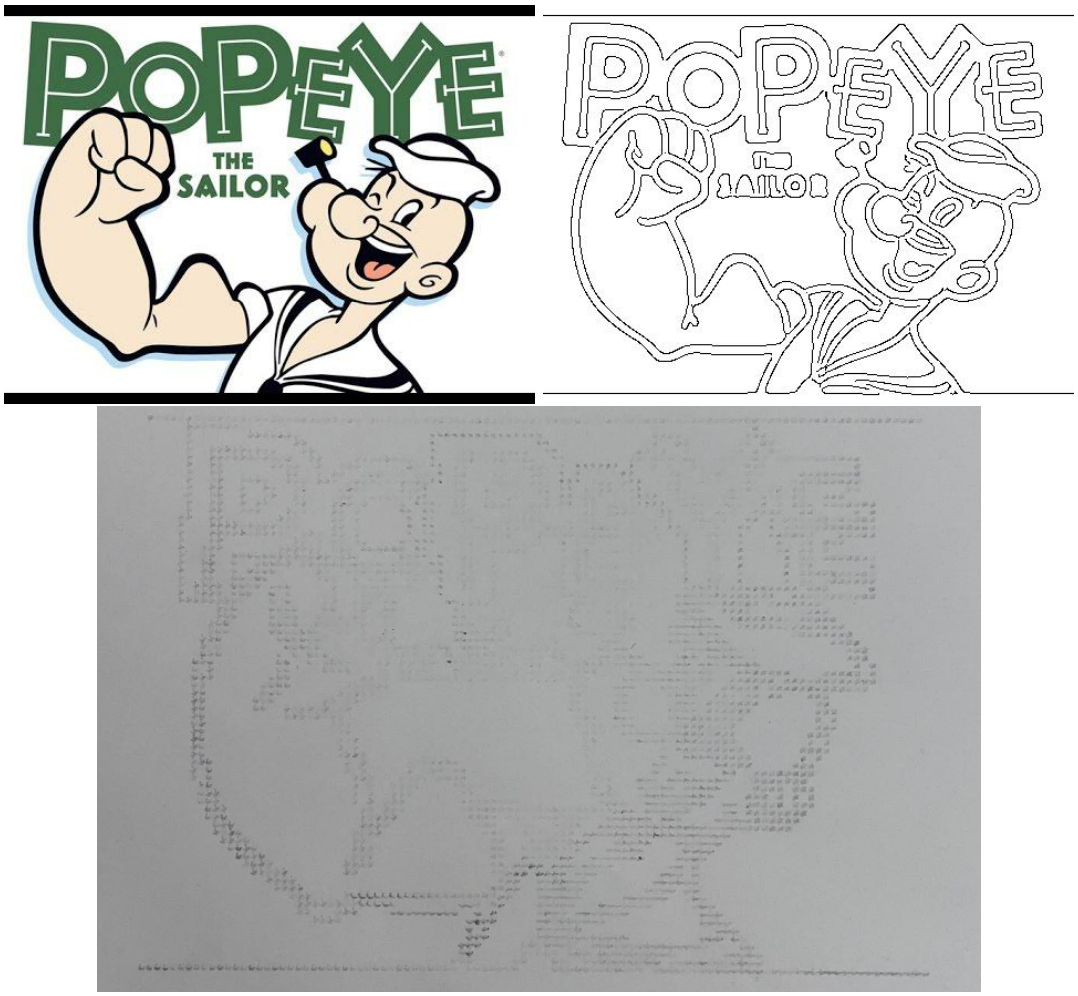

Figure 15: Picture type Popeye, from left to right: Imported picture of Popeye, Edge detected picture of Popeye, Drawing of Popeye

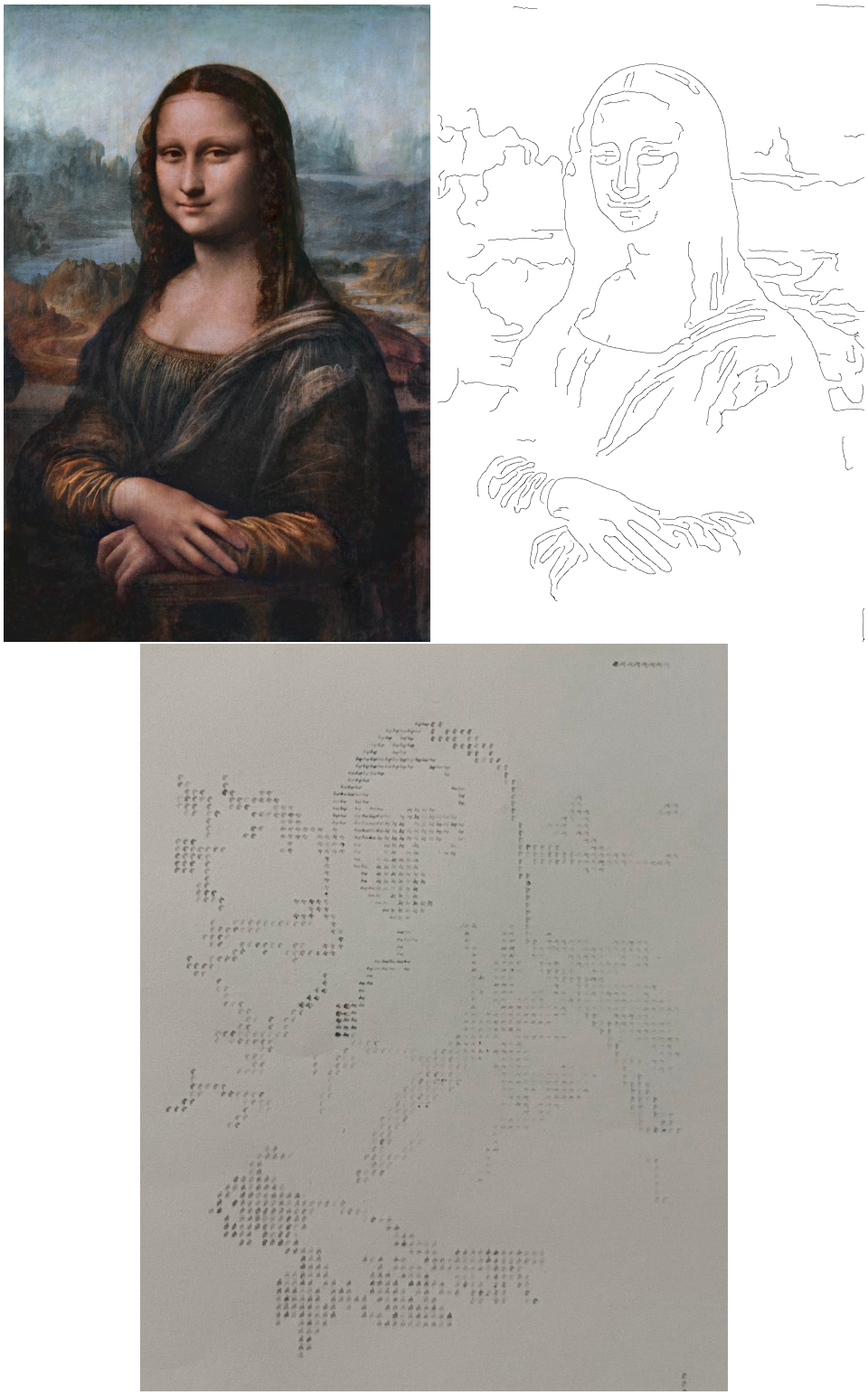

Figure 16: Picture type Mona Lisa, from left to right: Imported picture of Mona Lisa, Edge detect of Mona Lisa, Drawing of Mona Lisa

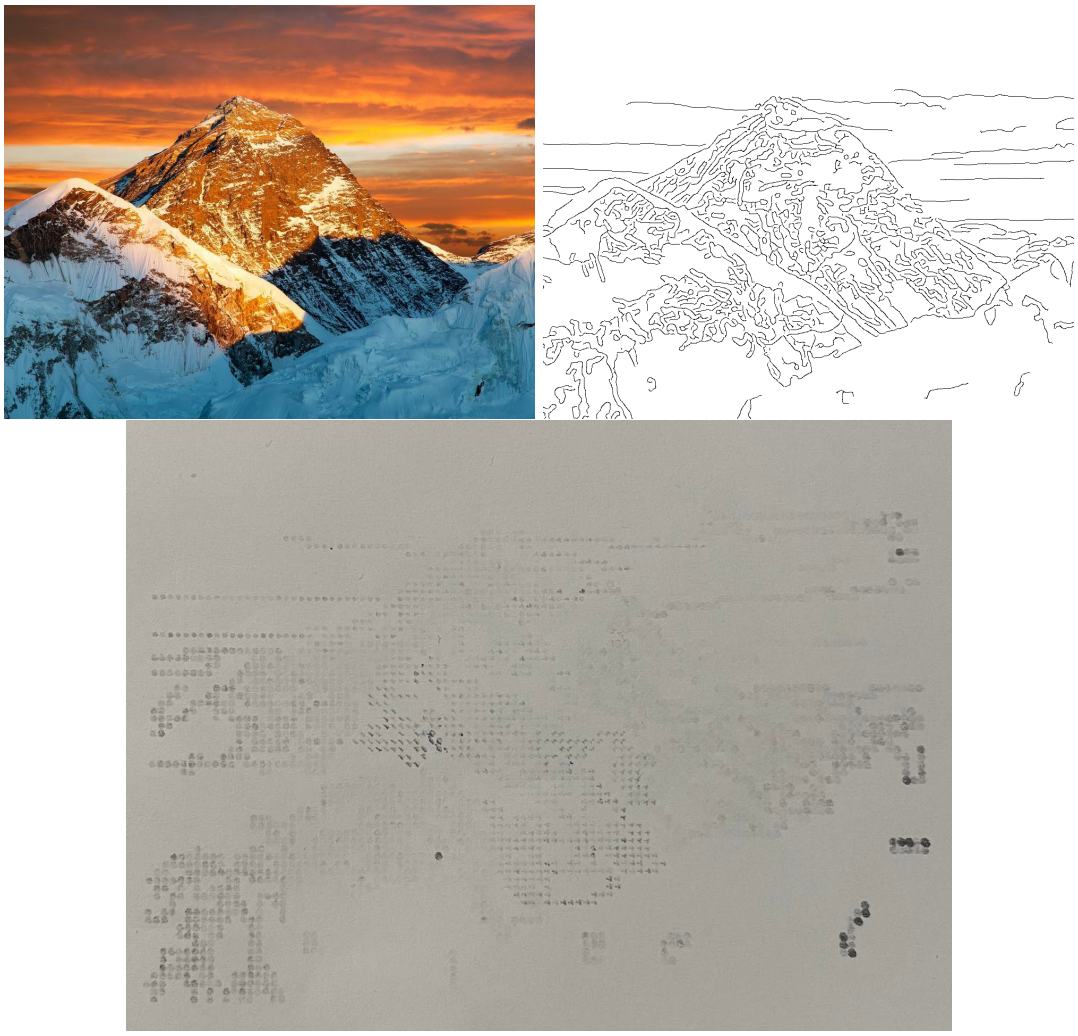

Figure 17: Picture type Mountain, from left to right: Imported picture of Mount Everest, Edge detect of Mount Everest, Drawing of Mount Everest

5.3 Experiment with picture resolutions

In this experiment the idea was to draw the same picture but with less resolution, to show the difference. The resolution of the pictures was changed by inbuilt Mathematica function ImageResize. The image used, had an EdgeDetect value of 12.

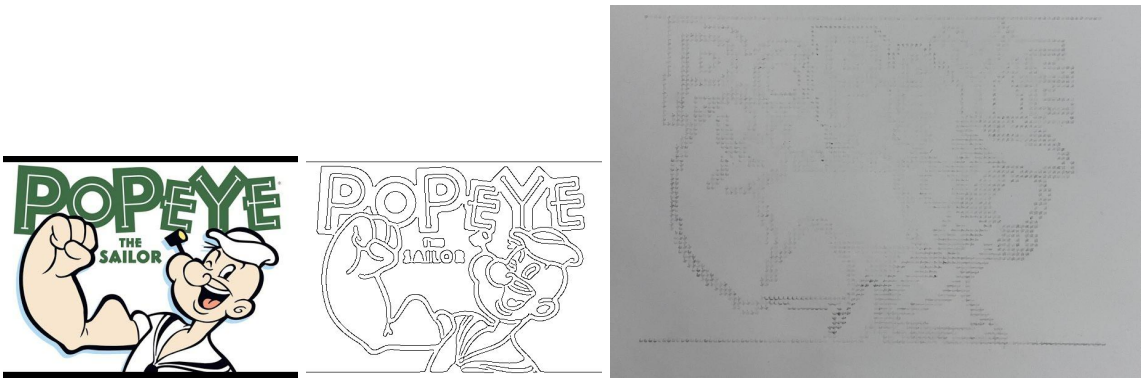

Figure 18: Popeye resolution : (480 x 360), drawing duration: 50 minutes

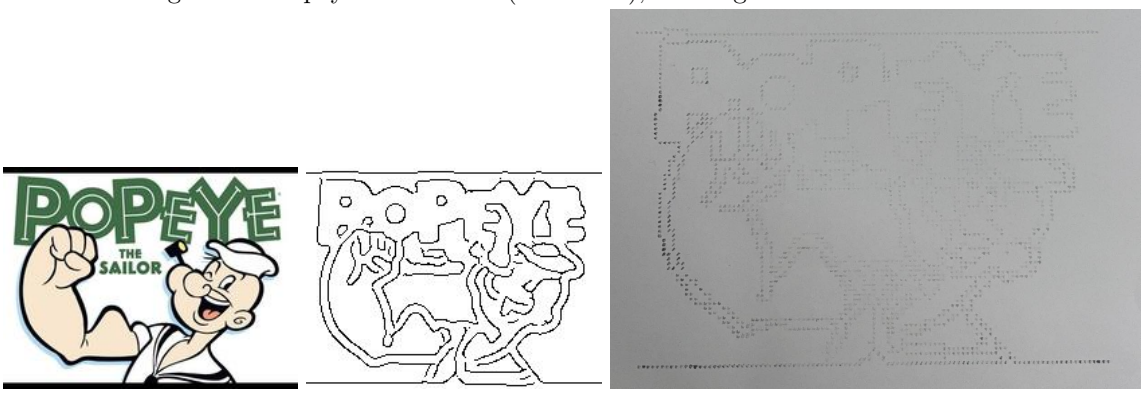

Figure 19: Popeye resolution : (220 x 165), drawing duration: 35 minutes

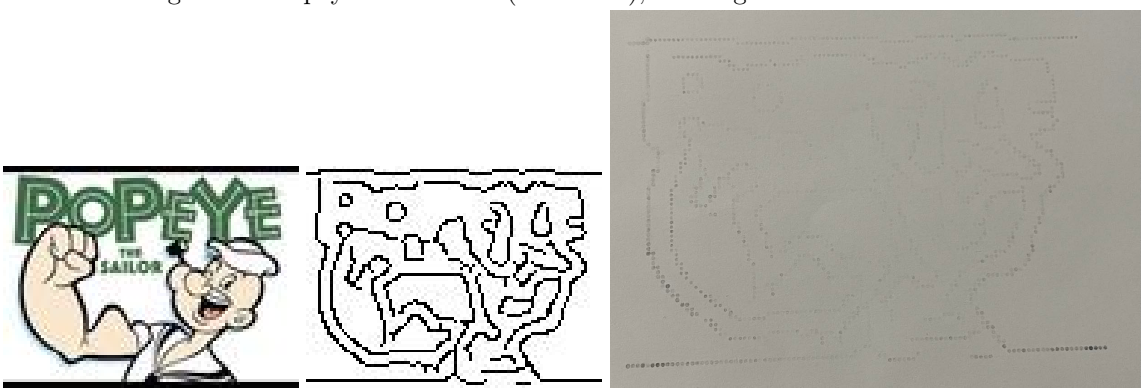

Figure 20: Popeye resolution : (100 x 75), drawing duration: 20 minutes

5.4 Partial discussion

Looking at the three pictures of Furby with variation in robot speed, see (fig. 13), no apparent difference can be seen in quality. The results are similar regardless of drawing speed, however with higher speed the robot will start to shake more than usual, which causes the robot to have increased pressure in some points in the drawing. But for a simple drawing like the furby, a lot of time can be saved with a higher drawing speed.

Throughout the picture type experiment it can be found that the final drawings match very well with the edgedetect pictures. All edges are sharp and the drawings are clear. (fig. 14). The second picture drawn was Popeye (fig. 15). The picture looks fine, due to it not having depth. The reason the points in the middle are unclear is due to a calibration issue with the robot. The third picture in this experiment was drawing the Mona Lisa (fig. 16). This drawing shows that the dot-method is not optimal for drawing pictures with depth. Her face and background merge together, thus making the drawing deviate too much from the original image.
The fourth picture was an image of Mount Everest (fig. 17). This drawing has both depth and shadow, and as seen on the drawn image, it does not work well with the dot method. The dot method cannot differentiate between dark and light colors; the details of the drawing merges together.

For the last experiment an image of Popeye was used. The first drawing with (480 x 360) resolution looked the best but required the most amount of time. (fig. 18). The second drawing with (220 x 165) resolution was a little unclearer, but it was still acceptable, and was less time-consuming than the first drawing (fig. 19). The third drawing with (100 x 75) resolution was the worst one. Not only was it unrecognizable from the original image, the details of the script, body and face were almost invisible (fig. 20). A correlation is found between resolution and the amount of detail; The smaller the dimensions the more details get sacrificed. For each individual drawing it seems there is a so called "sweet spot" where the drawing is clear and the time it takes the robot to draw is not absurdly high.

5.5 Partial conclusion

The experiment with speed, showed no substantial difference in drawing quality. Increasing the speed can therefore be a time saver.
The experiment with different types of pictures, showed that the dot method can be less effective when the pictures have a different depth in colors. It cannot make a distinction between colors or depth in a drawing. The dot method is more effective in drawing edges rather than the details of an image.
The experiment with resolution showed that the smaller the dimensions the more details get sacrificed. However, less points decrease the amount of time it takes the robot to draw. So there is "sweet spot" between time and resolution, when referring to the quality of the drawing.

6 Discussion

The image was analyzed in Mathematica, with its inbuilt functions. As for the edge detect function, which had different second parameter for different pictures, it was adjusted so the edges were clear and visible, see section 3.1. The drawing size of the robot was set from -50 to 50 in (x,y) coordinates. The reason for that was that the robot would not allow to exceed these values in coordinates. That means coordinate (50,51) was unreachable by the robot.

The pencil sharpener was designed to sharpen the pencil without a simple contact, not to be visually appealing. Therefore glue was used to hold the 2 infrared diodes in place, and made it quite easy to position the 2 because the glue did not solidify immediately, however if the positioning was decided too quick and it was incorrect then the glue would be solidified and it would have to be redone. Tape was tried as a solution in the beginning, but it was too hard to position the 2 diodes so that the infrared light would reflect correctly. It could be discussed whether a different kind of tape may have cause a different result, but this was not tested.

The circuit for the pencil sharpener was designed on a breadboard due to uncertainty about how the actual circuit was going to look like. The plan was to redesign the circuit(9.2) found online, and make it so that the LED that would turn on as infrared light is reflected, would instead be replaced by a motor turning on as infrared light is reflected. Therefore the breadboard made it easy to change components and wiring quick, rather than making a PCB-board and then have to redo it due to wrong components. The negatives of using the breadboard were that the length of the 4 wires leading to the infrared diodes and the length of the 2 wires leading to the motor all had to be quite long because the pencil sharpener had to be fit into the carved out drawing-platform of the robot while still being connected to the breadboard that was placed next to the robot on the table. This sometimes lead to confusion with wiring. See (section 4.2).

During experimentation, see section 5, it was discovered that there was an issue with the robot. The calibration was made properly, but some points were more visible than others. Observations of the drawing process showed that the robot was unstable, while drawing, at the edges of the drawing. The robot shook with the pencil, and as a consequence it made points at the edges of the drawing more visible than in the middle of the drawing.

7 Conclusion

The code made in Mathematica analyzes the imported image and converts it into G-code, which is then uploaded to the robot interface and the robot can start drawing. The method used for drawing in the project was "dot-method". The drawing dimensions in which the robot could operate was given in X and Y coordinates. The drawing dimensions for all pictures ranged from -50 to 50 in (x,y). The height of the pencil should decrease by 0.2 in Z values, see (fig. 6). The inbuilt Mathematica functions FindShortestTour and EdgeDetect efficiently speeds up the process of drawing with the "dot-method". The FindShortestTour function orders the coordinates so the travel distance from point to point is minimal. The number of points drawn is decided by the EdgeDetect function.

The pencil sharpener was divided into two parts, a voltage regulator and a pencil circuit, see section 4.1 and section 4.2. The Voltage was reduced to 6 Volt by using the voltage regulator circuit with the "LM7805" transistor. The pencil circuit was responsible for making a switch by the infrared emitter and receiver.

Experiments made in section 5, showed that the increase in robot speed does not worsen the quality of the image, see (fig. 13). Drawings of different types showed, see (section 5.2), that the "dot-method" does not work well with depth. Mona Lisa, Popeye and Furby pictures are visible but the drawing of the Mountain does not represent the imported image. Therefore the imported image cannot be too complex. The image resolution experiment showed that the process of drawing the same image could also be sped up by reducing the image resolution. Although it takes less time, the quality of the drawing worsens. However, this relationship between time and resolution can be optimized by finding the "sweet spot". Here it neither takes too long to draw nor sacrifices too much detail.

8 Appendix

The appendix is attached with the code in the zip-file.

1. 1N4002 diode (see datasheet: "1N4002.pdf")
2. 1N4148 diode (see datasheet: "1N4148.pdf")
3. BC639 NPN Transistor (see datasheet: "BC639.PDF")
4. LM2904 Operational amplifier (see datasheet: "LM2904.pdf)
5. LM7805 Voltage regulator (see datasheet: "LM7805.pdf")
6. OP505 IR-LED (see datasheet: "OP505.pdf")
7. OP593 Photo-LED (see datasheet: "OP593.pdf")

9 Bibliography

[9.1] Learn By Watch. (2015, April 6). Science Project 12 Volt to 6 Volt Using IC7805 [Video]. https://youtu.be/YxeJaUNWdA4

[9.2] Elonics - Electronics Projects on Breadboard. (2017, March 25). IR Proximity Sensor / Obstacle Detector circuit on Breadboard — LM358 Op-Amp projects [Video]. https://www.youtube.com/watch?v=qEMtCKfZOHwlist=LL