Dot Master

Braille printer

FREDRIKA ARDESTAM

SARA SOLTANIAH
Dot Master

Braille printer

FREDRIKA ARDESTAM
SARA SOLTANIAH

Bachelor’s Thesis at ITM
Supervisor: Nihad Subasic
Examiner: Nihad Subasic

TRITA-ITM-EX 2018:61
Abstract

Braille is a writing system that uses tactile dots in a predetermined order which, in relation to each other, represent different letters in the alphabet. This writing system makes it possible for people with visual impairments to take part of the written media. But the availability of home based braille printers is limited and these printers are often expensive. The purpose of this project is to investigate if it is possible to build a home based braille printer for a low cost using microcontrollers, and thereby making it more accessible to people with visual impairment. In order to achieve this, a prototype was built using an microcontroller together with stepper motors and a solenoid. These components were then controlled by code through user input and translated to required movements. Each switch case then calls a set of functions that activates the steppers and the solenoid in the order needed to get the desired character. In the time frame given, the project resulted in a prototype able to print out the input it was given. As for the cost of building your own Braille printer in comparison to buying one on the market highly depends on what processing machines are accessible.

Keywords: Braille, 3D printer, Arduino
Acknowledgements

We would like to thank all the people that have helped us to reach our goals of this project. First we like to thank Nihad Subasic for being an inspiring and helpful supervisor and examiner. We would also like to thank the assistants Johan Ehrenfors, Martin Gylling, Stanislav Minko and Ruben Svensson for all their help in the lab. Thanks to Staffan Qvarnström for all the help with ordering components and help with technical problems. Lastly we would like to thank the prototype assistants and Tomas Östberg for all the help with creating physical parts of the construction.

Fredrika Ardestam and Sara Soltaniah Stockholm, May 2018
# Contents

1 Introduction 1
   1.1 Background ................................................. 1
   1.2 Purpose ............................................. 1
   1.3 Scope ............................................ 2
   1.4 Method ............................................ 2

2 Theory 3
   2.1 Braille ........................................... 3
   2.2 3D-printer ......................................... 7

3 Demonstrator 8
   3.1 Mechanics ........................................... 8
      3.1.1 Embossing Mechanism ............................. 8
      3.1.2 Motion Mechanism ................................. 9
   3.2 Electronics ......................................... 10
      3.2.1 Motor ........................................... 10
      3.2.2 Microcontroller ................................. 11
      3.2.3 Motor shield .................................... 12
      3.2.4 Microstepping driver ............................ 12
      3.2.5 Limit Switch .................................... 13
      3.2.6 MOSFET (transistor) ............................ 13
   3.3 Software ............................................. 14

4 Results 15
   4.0.1 Printing ........................................... 15
   4.0.2 Cost Estimate ...................................... 16

5 Discussion and Future improvements 17

Appendices 18

A Arduino code 19

Bibliography 38
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A representation of the structure of a Braille cell.</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>The Swedish alphabet in Braille characters.</td>
<td>4</td>
</tr>
<tr>
<td>2.3</td>
<td>How numbers are written in braille.</td>
<td>4</td>
</tr>
<tr>
<td>2.4</td>
<td>How some punctuations are written in braille.</td>
<td>4</td>
</tr>
<tr>
<td>2.5</td>
<td>Dimensions of the Braille cell.</td>
<td>5</td>
</tr>
<tr>
<td>2.6</td>
<td>Picture of a slate and stylus, used for writing Braille.</td>
<td>6</td>
</tr>
<tr>
<td>2.7</td>
<td>Illustration of a 3D printer mechanism and its axes.</td>
<td>7</td>
</tr>
<tr>
<td>3.1</td>
<td>3D model of the constructions embossing mechanism, created in Solid Edge.</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Mechanism from the inside of the 3D model, created in Solid Edge.</td>
<td>9</td>
</tr>
<tr>
<td>3.3</td>
<td>Mechanism from the outside of the 3D model, created in Solid Edge.</td>
<td>10</td>
</tr>
<tr>
<td>3.4</td>
<td>Picture of the Arduino microcontroller.</td>
<td>11</td>
</tr>
<tr>
<td>3.5</td>
<td>Picture of the Adafruit Motor Shield.</td>
<td>12</td>
</tr>
<tr>
<td>3.6</td>
<td>Picture of the microstepping driver used to control one of the stepper motors.</td>
<td>13</td>
</tr>
<tr>
<td>4.1</td>
<td>The finished Braille printer</td>
<td>15</td>
</tr>
</tbody>
</table>
Nomenclature

CAD  Computer Aided Design
IDE  Integrated Development Environment
MOSFET  Metal Oxide Semiconductor Field Effect Transistor
MTM  Swedish Agency for Accessible Media
SEK  Swedish Krona
Chapter 1

Introduction

This chapter will discuss the basic idea of what is to be accomplished during this project. It will present the background, purpose, and scope of the project.

1.1 Background

Braille is a writing system that uses tactile dots in a predetermined order in relation to each other. Each letter in the alphabet has a corresponding combination of dots. This system makes it possible for individuals with visual impairment to read written media. The availability of Braille printed media is currently limited. Companies that print literature and other printed media in Braille do exist, although, there is only a small percentage of all printed media that get translated into Braille. A daily obstacle for individuals with visual impairments is that the mail they receive is not printed in Braille, the result is that they need to rely on others to be able to access the information they have received. By creating a home-based printer which can scan regular text and print it out in Braille, the everyday lives of these individuals could get easier and it may give them a stronger sense of independence and let them be part of the written culture.

1.2 Purpose

The purpose with this project was to investigate if it is possible to create a home-based printer that is able to take text input from a computer, and based on this, print the text out in Braille. To find an answer to this question a machine with these abilities was built. Such a machine could facilitate the everyday life of an individual living with visual impairment.
1.3 Scope

The main goal of this project was to construct a machine, that with given text input from the user, print out it in Braille. When this was achieved the cost of constructing the printer was compared to an already existing printer on the market, to see if it was possible to create a cheaper printer. In the process of reaching this, the following research questions had to be answered:

1. Can microcontrollers be used to build a machine that can facilitate the par-taking of written media for the visually impaired?

2. Can a construction be made that cost less than the already existing solutions?

Braille characters are small and the upraised dots in a character have a diameter of 1 mm, therefore, the process of creating indentations requires small tools. The spike that pushes the paper down into the underlay would then have to have a diameter of 1 mm. With the machines available, it was difficult to create durable parts with such small dimensions, while still keeping high precision.

1.4 Method

To be able to answer the research questions stated in the section above, a demonstra-tor was built. The CAD program Solid Edge [1] was used to create a computerized 3D model of the construction and all its components. With the help of 3D printers and laser cutters the different components were thereafter made. An Arduino Uno Rev3 [2] microcontroller board and an Adafruit motor shield v2.3 [3] in combina-tion with software written for the microcontroller were used to control the electrical components, such as stepper motors, end switches and a solenoid.
Chapter 2

Theory

This chapter will discuss the theory behind the two main concepts of this project, the Braille writing system and the basics behind 3D-printers moving mechanisms.

2.1 Braille

Braille is a writing and reading system, where the readers run their fingers over a set of ordered raised dots. This system was developed for individuals with visual impairment to make it possible for them to read texts. The Swedish authority, Swedish Agency for Accessible Media (MTM) prints about 450 books in Braille every year. The selection of books to be printed by MTM is chosen based on suggestions given by Braille readers, MTM also chooses a few new fiction and nonfiction books to print. [4]

Each letter or symbol has its own combination of dots in Braille. This means that Braille is not a language by its own, but rather, a system to make written information accessible for individuals with visual impairment. Therefore, it is possible for different languages, to use the same system when converting regular text into tactile text. [5]

Braille characters consists of a rectangular area, called a cell, in which there can be up to six raised dots in different formations. Each dot in the cell has its own number, that is shown in Figure 2.1 below.

Figure 2.1. A representation of the structure of a Braille cell. [6]
CHAPTER 2. THEORY

Up to 63 different dot combinations can be achieved. These combinations constitute, among other, the letters, numbers and punctuations that are used in regular text. The Swedish alphabet written in Braille is presented in Figure 2.2 below.

Figure 2.2. The Swedish alphabet in Braille characters. [6]

When writing numbers, the same dot combinations as for letters A-J (numbers 0-9) are used with the addition of a special number cell which is written before the set of letters corresponding to a set of numbers, see Figure 2.3 below. The Braille characters that represent punctuations are shown in Figure 2.4.

Figure 2.3. How numbers are written in braille. [7]

Figure 2.4. How some punctuations are written in braille. [8]
2.1. BRAILLE

Braille have standardized measurements, in Sweden each cells dimension is 3,5x6 mm. The internal dimensions may vary in different countries but the most common are as follows:

- The dot diameter is 1 mm and within the cell.
- Dots are placed 2,5 mm from one another, measured from the center points of the dots, both horizontally and vertically.
- The height of which the dot stands out from the paper is 0,25 mm.
- The different cells are placed beside one another horizontally by a distance of 6 mm and the vertical distance is 10 mm, which is also measured from the center points.

The different cells are placed beside one another horizontally by a distance of 6 mm and the vertical distance is 10 mm, which is also measured from the center points, see Figure 2.5 Braille standards.

Figure 2.5. Dimensions of the Braille cell. [9]
When writing Braille by hand, a stylus and a stale are used, see Figure 2.6. The stale has a bottom plate with hemispherical indentations and a top part that frames the Braille cells. The bottom part and the top part are joined with a hinge, and the paper is placed between the top and bottom part when writing. The stylus is a tool used by the writer to make raised dots by pressing the stylus down into the indentations.

Figure 2.6. Picture of a slate and stylus, used for writing Braille. [10]

When writing Braille, it is important to remember that whatever is written will be mirrored. Therefore, it should be written from right to left with mirrored characters, so that the side with elevated dots will always display a fully readable text.

The printer also needs some sort of stylus that makes the indentations. [11]
2.2 3D-printer

3D printers have in recent years become very popular. They work in the way of layering molten material in cross sections upon each other enough times to create a three-dimensional model. [12]

This project takes inspiration from 3D printers when creating the mechanism that positions the Braille stylus. With a few relatively simple mechanical movements the 3D printer can print an almost unlimited variety of shapes. Most 3D printers have arms in a three axes system to move the printing head. The X-axis moves the head right and left, the Y-axis moves it back and forth and lastly the Z-axis moves the build plate up and down, see figure 2.7 below. In this project only the X- and Y-axis will be used.

Figure 2.7. Illustration of a 3D printer mechanism and its axes. [13]

A common way to drive the arms of a 3D printer is by using stepper motors. Stepper motors are relatively low priced and are only required to run at a low speed in 3D printing applications. Thus they require a low level of feedback. In combination with a stepper driver the stepper motor can be controlled to output a small rotation and thus precisely achieve the desired motion. [14]
Chapter 3

Demonstrator

This chapter will take a look at the construction built during this project. It will present the components necessary to build a functioning Braille printer.

3.1 Mechanics

This section will present and explain the components and functions of the embossing and motion mechanism.

3.1.1 Embossing Mechanism

The embossing mechanism creates the indentations of the Braille letters, using a linkage between a solenoid and a nail. Since the core of the solenoid gets pulled into the coil when energized, the nail should not be directly connected to the core. To make the nail push down into the paper, there needs to be an arm between the solenoid core and the nail, see Figure 3.1 below how this was constructed. The linking arm has three holes in which axes will be placed to keep the parts connected and allow rotation. The two outer ones are extended to allow the rotating movement when the core is pulled into the solenoid. One end of the arm is connected to the core and the other end is connected to the nail holder. This way the nail is able to press down into the paper when the core is pulled up into the solenoid. When writing the code for the printer it is important to remember that the letters need to be printed in a mirrored manner.
3.1. MECHANICS

Figure 3.1. 3D model of the constructions embossing mechanism, created in Solid Edge. [1]

3.1.2 Motion Mechanism

Taking inspiration from a 3D printer the motion mechanism transports the embossing mechanism using two rotating belts moving in perpendicular directions (along an x- and y-axis), as can be seen in Figure 3.2. For each belt there are two metallic rods that act as tracks for the mechanism respectively the platform it is attached to. In order to move the platform and the embosser, the belts sit on pulleys that rotates with the help of stepper motors, one for each direction. To lower the friction between the rods and the platform respectively the embosser, brass bearings were made.

Figure 3.2. Mechanism from the inside of the 3D model, created in Solid Edge. [1]
In between the vertical metallic rods and the horizontal rods the bottom plate will be placed and on top of that an embossing plate will be placed, which can be seen in the figure below.

**Figure 3.3.** Mechanism from the outside of the 3D model, created in Solid Edge.

The embossing plate needed to be elastic so the indentations created in the paper from the nail would be better presented.

### 3.2 Electronics

This section will discuss the electrical components necessary to get the printer working.

#### 3.2.1 Motor

To get the movement of the embossing mechanism two stepper motors was used, one for each direction. The stepper motor used in this project is the 2-phase motor Tamagawa TS3214N61 with six wires. The stepper motor allows the motor axis to rotate in small steps which makes it possible to get more precise movements out of the motor. It is also possible for the motor to rotate the axis both forward and backwards with the help of a H-bridge. This specific motor has 200 steps in a full rotation, that means that the axis rotates 1.8 degrees per step. With this information in combination with the diameter of the pulleys it is possible to calculate the distance the belt moves for a specific number of steps.

\[
d = \frac{\alpha}{360} \times 2\pi r \tag{3.1}
\]

Where \(d\) is the distance the belt moves in mm, \(r\) is the outer radius from the center of the motor axis to the outer radius of the pulley in mm and \(\alpha\) is the rotation angle in degrees. This formula can then be used to calculate the number of steps needed to move the desired distance.
3.2. ELECTRONICS

3.2.2 Microcontroller

The Arduino Uno microcontroller [2] was used to manage the different electrical components such as stepper motors, limit switches, and the solenoid. With the help of code written in with Arduino software IDE [16], the microcontroller controls the flow of electricity through the different components so that the right component gets energized at the right time.

Figure 3.4. Picture of the Arduino microcontroller. [2]
3.2.3 Motor shield

In order to drive the motors both back and forth, with a specific number of steps and a certain speed, a Motor Shield was used. The one used in this project was an Adafruit motor shield v2.3, shown in figure 3.5 below [3], with the capabilities of driving up to two stepper motors. Aside from driving the stepper motors it is also able to energize the limit switches and the solenoid.

Figure 3.5. Picture of the Adafruit Motor Shield. [3]

3.2.4 Microstepping driver

Since one of the motor outlets in the motor shield broke during testing, a new way to control one of the motors was needed. In the end a microstepping driver A4988 was used to replace the the broken motor outlet. The A4988 is a microstepping driver that drives bipolar stepper motors. The driver has a built-in translator which makes it possible to control the number of steps with one pin and rotation direction with another pin. It also has a potentiometer for adjusting the current output, crossover-current protection and over temperature thermal shutdown. Maximum operating voltage is 35 V and minimum is 8 V. Logic voltage is from 3 to 5.5 V and if cooling is provided the maximum current phase is 2 A otherwise it can only handle 1 A. The driver has 16 pinouts. The two pins VDD and GND on the right bottom side gives the motor driver power, in our case it came from the Arduino board which provided 5 V. The four pins 1B, 1A, 2A and 2B is for connecting the motor, where 1A and 1B are for one coil of the motor and 2A and 2B are for the other coil. The last two pins VMOT and GND on the right side are for powering the motor. These two needs to be connected to power supply from 8 to 35V but also to a decoupling capacitor with at least 47 µF for protecting the driver board, see figure 3.6.
3.2. ELECTRONICS

Figure 3.6. Picture of the microstepping driver used to control one of the stepper motors. [17]

The two pins, DIR and STEP on the left bottom side, controls the motor. DIR is the direction pin and controls which direction the motor will spin, it needs to be connected with a digital pin on the Arduino board. STEP controls how many steps the motor will take, for each pulse the motor takes one step. The SLEEP pin above puts the board in sleep mode, meaning that when the motor is not in use it will minimize the power consumption. The following pin RESET will set the translator to a predefined Home state. These are initial positions for the motors start position and will be different depending on what microstep resolution is used. The three following pins MS1, MS2 and MS3 are for choosing the step resolution. Lastly the ENABLE pin is for turning on and off the FET outputs, which is a translator that controls the electrical behavior of the motor driver by using electric field.

3.2.5 Limit Switch

To make sure that the moving mechanism would not try to go further than the tracks allows, limit switches were used. These were wired so that there is current flowing through them while their open. When the moving mechanism begins to move to far, a switch is activated, which in turn will trigger the Arduino to stop the motors.

3.2.6 MOSFET (transistor)

A transistor is a component with semiconducting properties. It can be used to control, regulate and amplify electrical signals. The basic principle of a transistor is that it typically has 3 electrical leads which are each called source, drain and gate. When supplying the gate with an electrical signal the conducting abilities of the semi conductive material (the substrate), often flowing between source and drain, is affected. [18] The current is provided from an external power source, for example a battery, while its flow rate through the transistor is controlled by signals given
to the gate. One type of transistor, the one used in this project, is the MOSFET which stands for Metal Oxide Semiconductor Field Effect Transistor. As the name implies it is a type of field-effect transistor, in other words it controls the behaviour of the device with the help of an electric field. These transistors can be categorized into two types, N-type or P-type. In a N-type transistor the source and drain are of N-type and the substrate is of P-type. The opposite applies to the P-type transistor. If for example the substrate is of N-type it means that it has a number of free electrons throughout the semiconductor. For P-type there is instead a lack of electrons resulting holes in the silicon lattice. [19] This means that for a N-type MOSFET, when applying a voltage to the gate, the current will be flowing through from drain to source. For the P-type MOSFET everything works in the opposite direction.

3.3 Software

In this project Arduinos library was used to write the code for all the functions of the construction. The code will first of all home the embossing mechanism by making the steppers rotate towards the origin until the embossing mechanism activates the limit switches. Thereafter, the Arduino takes text input from the input slot in Arduinos serial monitor, and based on that, enters the corresponding switch case, which activates certain print functions that makes the motors and solenoids move.
Chapter 4

Results

This chapter will discuss the results achieved at the end of the project and aims to give answers to the research questions stated in the scope.

4.0.1 Printing

The final construction can be seen in Figure 4.1 below. The construction is able to print out 66 Braille letters in one run-through, in other words without going through the homing process again. The mechanism places the dots at the right distances in relation to the other dots in the same letter but the depth of each dot is a bit smaller than the standard.

Figure 4.1. The finished Braille printer
The bottom plate of the model is required to be a stiffer material. In our model a 3 mm aluminum plate was used, after realizing that a 3 mm acrylic plastic sheet was too elastic which made the bottom plate deform too much and that resulted in less indentations.

4.0.2 Cost Estimate

When looking at the price ranges of existing home based Braille printers it often starts around 20 000 SEK. The cost estimate for the material used in the construction created in this project follows below.

- 3D printer filament 0.283 kg PLA: a total of 199 SEK for a 1 kg roll.[20]
- Aluminium sheet 340x295x3 mm: 224 SEK for 500x500x3 mm. [21]
- Acrylic plastic sheet 300x165x3 mm: 80 SEK. [22]
- Pulling solenoid with rated voltage 24V: 470 SEK. [23]
- Limit switches: 110 SEK (total for four switches)[24]
- Adafruit motorshield: 175 SEK. [3]
- 5 mm in diameter steel, total length 966 mm: 24 SEK (length 1 500 mm). [25]
- Timing belt: 90 SEK (length 1 000 mm). [26]

This leads to a total cost of 1 582 SEK. Adding an additional amount of approximately 1 000 SEK for electrical wires and associated components to make circuits and also screws and nuts, the cost of building would still be less than buying a new printer on the market. It is however necessary to take in consideration the cost and access of machines such as 3D printers, laser cutters and so on. Just including the water cutter, used to make the metal plate, in the cost would bring the total cost up to well above the price of a new Braille printer. The laser cutter almost reaches up to the price of a Braille printer and a 3D printer would cost a few thousand Swedish kronor as well.
Chapter 5

Discussion and Future improvements

The calculations in the results show that the cost of building the Dot Master printer would be less than buying a new printer on the market. However, it is necessary to consider that, the calculation is based on the cost for one printer made with access to expensive machines, such as 3D-printers and laser cutters, and not for mass production and standardized components. Making the Dot Master printer shows that it is possible to develop a printer cheaper than the ones on the market, by standardizing components, change manufacturing method, and mass produce.

A lot of problems around the movement mechanism occurred, since the axes were not perfectly parallel. The result was a lot of friction between the axes and the bushings, and as a consequence, the stepper motors had to reach a higher torque to make the mechanism move.

One of the solutions to this problem was to be very precise in the process of making the axes and their attachments. That way it is easier to achieve parallel axes. Thereafter it would be better to make all the attachment plates in metal, to make it sturdier and keep it from bending or in other ways change shape during the movements. The same goes for the axes attachments which were easily destroyed after too much screwing. Another possible solution would be to use bushings that are able to hold the lubricating oil better. The bushings used in this project did not have any play to keep the oil in. In the beginning the movement was smooth but the effect started decreasing fast since the oil started to gather at the ends instead. A final solution to solve the problems in the X-direction was to increase the torque by changing step type from microsteps to double. Microsteps give a much smoother movement, but with double, a higher torque was achieved by having two coils activated at the same time.

There were also some problems with the embossing mechanism. It proved difficult to achieve proper indentations for the braille text. Part of the problem was the bottom plate, on which the embossing plate was placed on. It was made with a three millimetre thick acrylic plastic sheet, that easily gave way when the nail pressed down into the embossing plate. Therefore, some of the force, needed to make the indentations, where instead taken up by the bottom plate. To solve this a
new bottom plate was made in aluminum. The new plate did not bend like the old one, but still the indentations were not deep enough. Another problem turned out to be that the embossing plate did not have enough give. The first plate was 3D printed with a flexible filament called NINJAFLEX [27]. This turned out to be too hard for the mechanism to press down the nail into. Different foam materials were then tested with the mechanism and in the end, it seemed that the type of foam at the back of old computer mouse pads was a well suited material. With this material it was possible to achieve indentations that were possible to both see and feel, but they still did not reach the standard dimensions of a braille dot. By recalculating the dimensions of the linkage arm, and making sure that the mechanism cannot shift upwards when embossing, and getting a sharper nail, it should be possible to achieve the correct dimensions.
Appendix A

Arduino code
APPENDIX A. ARDUINO CODE

```cpp
/*
 * Code for Domaster
 * This code is for the Domaster machine.
 * It will first move the mechanism to home position with help of the limit switches,
 * then wait for input from the Serial monitor by the user
 * and lastly make a movement based on the given input
 * 
 * Observe! Make sure the Serial monitor is set on new line, before entering the text
 * you want to print.
 * 
 * Name: Sara Soltanash and Fredrika Ardestam
 * Programme: Design and Product Realisation (CDEPR)
 * Date: 29-08-2013
 * University: KTH
 * Course: MFL15K- Degree Project in Mechatronics, First Cycle
 * 
 */

#include <Wire.h>
#include <Adafruit_MotorShield.h>

// Defines pins numbers
int Xswitch = 3;  // for limit switch X, needs to be pin 2 or 3 when using interrupts
                  // on arduino uno
int Yswitch = 2;  // for limit switch Y
int solenoid = 9; // for solenoid
const int stepPin = 5;  // for motor V
const int dirPin = 4;   // for motor V, changes the rotations direction
int sleepTime;        // for motor V, makes it possible to minimize power consumption when
                      // the motor is not in use.

// Global variables
volatile int SwitchXFlag;  // Says that this integer can switch at any time, will be using
volatile int SwitchYFlag;  // --||--
d int index = 0;  // index for each character input
char OneCharInput;      // each character will be saved before printing it out
char inputString[100];  // will store all the characters in this together
char EndMarker = '\n';  // for compare the input with endmarker to know if has reached
                        // the end character or if there is more
boolean newData = false; // Stops the while-loop in ReadInput function which reads in all
                          // the characters one by one when newData == true;
int distance;           // inputs for moveInX and moveInY functions, the distance
```
```cpp
// Input is the standard space for Braille

int stepsCounterX; // Counts how many steps has been taken so the Arduino knows how
// many to take in reverse when it comes to the end (horizontal)
int stepsCounterY; // == -1 == (vertical)

// For the Adafruit MotorShield v2.8
Adafruit_MotorShield AHMS = Adafruit_MotorShield(); // Create the motor shield object
// with the default I2C address
Adafruit_StepperMotor *StepperX = AHMS.getStepper(200, 1); // Connect a stepper motor with 200...
// ...steps per revolution (1.8 degree), to motor port #1 (A1 and A2)
Adafruit_StepperMotor *StepperY = AHMS.getStepper(200, 2); // will not be using because this side...
// ...if the shield is broken, for this motor a 4888 motor driver will be used

// functions for homing the machine vertical (X) and horizontal (Y)

void backwardStepX()
#

StepperX->step(2, BACKWARD, DOUBLE); // takes 2 steps, in reverse with double stepping

void backwardStepY()
#

digitalWrite(sleep, HIGH); // turns the motor on
digitalWrite(disPin, LOW); // Changes the rotation direction, LOW is up
// Halts 400 pulses for making two full cycle rotation
for(int x = 0; x < 2*16; x++) // the for-loop will make it possible for stepping by setting the pin
// high and low, '16 is because microsteps has sixteen step resolution
{
  digitalWrite(stepPin, HIGH);
  delayMicroseconds(600);
  digitalWrite(stepPin, LOW);
  delayMicroseconds(600);

  // delay(1000);
}

// function to make movement in X direction. It needs a distance input which can be: 3.5, -3.5 or 6.0
void moveInX(float distance)
#

if (distance == 2.6) // moves 2.6 mm in writing direction (direction: left)
{
  StepperX->step(15, FORWARD, DOUBLE);
  stepsCounterX=stepsCounterX+1;
}

if (distance == -2.5) // moves 2.5 mm in reverse writing direction (direction: right)
```
APPENDIX A. ARDUINO CODE

```cpp
{
  StepperX->step(15, BACKWARD, DOUBLE);
  stepsCounterX=stepsCounterX-15;
}
if (distance==6.0)  //moves 6.0 mm, for new letter, in writing direction (direction: left)
{
  StepperX->step(6, FORWARD, DOUBLE);
  stepsCounterX=stepsCounterX+6;
}
}   // Function to make movement in Y direction. It needs a distance input which can be: 2.6, -2.6 or 10.0

void moveY(float distance)
{
  digitalWrite(e1pwm, HIGH); // turns the motor on
  if (distance==-2.6)  //moves 2.6 mm in reverse writing direction (direction: up)
  {
    digitalWrite(dirPin, LOW); //Changes the rotation direction, LOW is up
    for(int x = 0; x < 15*16; x++) // the for-loop will make it possible for stepping by setting the pin high
    // and low, *16 is because microsteps has sixteenth step resolution
    {
      digitalWrite(stepPin, HIGH);
      delayMicroseconds(800);
      digitalWrite(stepPin, LOW);
      delayMicroseconds(800);
    }
    stepsCounterY=stepsCounterY-15;
    delay(2000);
  }

  if (distance==2.6)  //moves 2.6 mm in writing direction (direction: down)
  {
    digitalWrite(dirPin, HIGH); //Changes the rotation direction, HIGH is down
    // Makes 200 pulses for making one full cycle rotation
    for(int x = 0; x < 15*16; x++) // the for-loop will make it possible for stepping by setting the pin high
    // and low, *16 is because microsteps has sixteenth step resolution
    {
      digitalWrite(stepPin, HIGH);
      delayMicroseconds(800);
      digitalWrite(stepPin, LOW);
      delayMicroseconds(800);
    }
    stepsCounterY=stepsCounterY+15;
  }
```
delay(1000); // One second delay
}

if (distance==10.0) // moves 10.0 mm in writing direction (direction: down), for a new ro
{
  digitalWrite(disPin, HIGH); // Changes the rotation direction, HIGH is down
  // Makes 200 pulses for making one full cycle rotation
  for(int x = 0; x < 50*16; x++) // the for-loop will make it possible for stepping by setting the pi
  /* It is because microsteps has sixteenth step resolution*/
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(850);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(850);
  } // stepsCounterY=stepsCounterY-88;
  delay(1000); // One second delay
  digitalWrite(sleep, LOW); // turns off the motor
}

// not using this because the shield is not working on this side
/
{
  StepperY->step(16, FORWARD, MICROSTEP);
}

if (distance==2.5) // GOING UP
{
  StepperY->step(16, BACKWARD, MICROSTEP);
}

if (distance==10.0) // GOING DOWN, for new row
{
  StepperY->step(88, FORWARD, MICROSTEP);
}

void solenoidfunc()
{
Serial.println("solenoid"); // Prints the message in serial monitor
  delay(1000); // wait 1 second
  digitalWrite(solenoid, HIGH); // activates the solenoid by giving it a high voltage level
  delay(1000); // wait 1 second
void PrintLetter(char character) {
    int upper = character - 'A';
    int lower = character - 'a';
    for (int i = 0; i < 6; i++) {
        if (upper < 6 && lower < 6) {
            solenoidfunc();
            moveInY(2.5); // moving back to start position in cell
            moveInX(2.5); // moving horiztonal for a new letter
        } else if (character < 32) { // convert the capital letter to a lowercase letter, ASCII
            character += 32;
        }
        if (upper > 1) {
            moveInY(2.5);
        } else if (lower > 1) {
            moveInX(2.5);
        } else if (character == 32) {
            break;
        } else if (character == 97) {
            break;
        } else if (character == 122) {
            break;
        } else if (character == 65) {
            break;
        } else if (character == 90) {
            break;
        } else if (character == 66) {
            break;
        } else if (character == 91) {
            break;
        } else if (character == 67) {
            break;
        } else if (character == 92) {
            break;
        } else if (character == 68) {
            break;
        } else if (character == 93) {
            break;
        } else if (character == 69) {
            break;
        } else if (character == 94) {
            break;
        } else if (character == 70) {
            break;
        } else if (character == 95) {
            break;
        } else if (character == 71) {
            break;
        } else if (character == 96) {
            break;
        } else if (character == 72) {
            break;
        } else if (character == 97) {
            break;
        } else if (character == 73) {
            break;
        } else if (character == 98) {
            break;
        } else if (character == 74) {
            break;
        } else if (character == 99) {
            break;
        } else if (character == 75) {
            break;
        } else if (character == 100) {
            break;
        } else if (character == 76) {
            break;
        } else if (character == 101) {
            break;
        } else if (character == 77) {
            break;
        } else if (character == 102) {
            break;
        } else if (character == 78) {
            break;
        } else if (character == 103) {
            break;
        } else if (character == 79) {
            break;
        } else if (character == 104) {
            break;
        } else if (character == 80) {
            break;
        } else if (character == 105) {
            break;
        } else if (character == 81) {
            break;
        } else if (character == 106) {
            break;
        } else if (character == 82) {
            break;
        } else if (character == 107) {
            break;
        } else if (character == 83) {
            break;
        } else if (character == 108) {
            break;
        } else if (character == 84) {
            break;
        } else if (character == 109) {
            break;
        } else if (character == 85) {
            break;
        } else if (character == 110) {
            break;
        } else if (character == 86) {
            break;
        } else if (character == 111) {
            break;
        } else if (character == 87) {
            break;
        } else if (character == 112) {
            break;
        } else if (character == 88) {
            break;
        } else if (character == 113) {
            break;
        } else if (character == 89) {
            break;
        } else if (character == 114) {
            break;
        } else if (character == 90) {
            break;
        } else if (character == 115) {
            break;
        } else if (character == 91) {
            break;
        } else if (character == 116) {
            break;
        } else if (character == 92) {
            break;
        } else if (character == 117) {
            break;
        } else if (character == 93) {
            break;
        } else if (character == 118) {
            break;
        } else if (character == 94) {
            break;
        } else if (character == 119) {
            break;
        } else if (character == 95) {
            break;
        } else if (character == 120) {
            break;
        } else if (character == 96) {
            break;
        } else if (character == 121) {
            break;
        } else if (character == 97) {
            break;
        } else if (character == 122) {
            break;
        } else {
            character = 32;
        }
        solenoidfunc();
        moveInX(2.5);
        moveInY(2.5);
        moveInX(-2.5);
        moveInY(-2.5);
        moveInX(-2.5);
        moveInY(2.5);
        moveInX(2.5);
        moveInY(-2.5);
        mov
    }
}

void ServoControl(int pin, float angle, float speed) {
    analogWrite(pin, angle * speed);
}

void setup() {
    Serial.begin(9600);
    pinMode(solenoidPin, OUTPUT);
    digitalWrite(solenoidPin, HIGH);
    delay(1000);
    digitalWrite(solenoidPin, LOW);
    delay(1000);
}

void loop() {
    if (Serial.available()) {
        char character = Serial.read();
        Serial.print(character);
        PrintLetter(character);
    }
}

int main() {
    Serial.begin(9600);
    while (true) {
        char character = Serial.read();
        Serial.print(character);
        PrintLetter(character);
    }
    return 0;
}
case 'd':
  solenoidfunc();
  moveInX(2.0);
  solenoidfunc();
  moveInY(2.8);
  solenoidfunc();
  moveInX(-2.5); // moving back to start position in cell
  moveInY(-2.5); // moving back to start position in cell
  break;
  case 'e':
    solenoidfunc();
    moveInX(2.0);
    solenoidfunc();
    moveInX(-2.5); // moving back to start position in cell
    moveInY(-2.5); // moving back to start position in cell
    break;
  case 'f':
    solenoidfunc();
    moveInX(2.0);
    solenoidfunc();
    moveInX(-2.5); // moving back to start position in cell
    moveInY(2.8);
    solenoidfunc();
    moveInY(-2.8); // moving back to start position in cell
    break;
  case 'g':
    solenoidfunc();
    moveInX(2.0);
    solenoidfunc();
    moveInX(-2.5); // moving back to start position in cell
    moveInY(2.8);
    solenoidfunc();
    moveInY(-2.8); // moving back to start position in cell
    break;
  case 'h':
    solenoidfunc();
    moveInY(2.8);
    solenoidfunc();
    moveInX(2.0);
    solenoidfunc();
    moveInX(-2.8); // moving back to start position in cell
    moveInY(-2.8); // moving back to start position in cell
    break;
APPENDIX A. ARDUINO CODE

```c
271     case 'a':
272         moveInX(2.5);
273         solenoidfunc();
274         moveInX(-2.5);  // moving back to start position in cell
275         moveInY(2.6);
276         solenoidfunc();
277         moveInY(-2.5);  // moving back to start position in cell
278         break;
279     case 'b':
280         moveInX(2.5);
281         solenoidfunc();
282         moveInY(2.6);
283         solenoidfunc();
284         moveInX(-2.5);  // moving back to start position in cell
285         moveInY(-2.5);  // moving back to start position in cell
286         break;
287     case 'c':
288         solenoidfunc();
289         moveInY(2.6);
290         solenoidfunc();
291         moveInY(2.6);
292         solenoidfunc();
293         moveInX(-2.5);  // moving back to start position in cell
294         moveInY(-2.5);  // moving back to start position in cell
295         break;
296     case 'd':
297         solenoidfunc();
298         moveInY(2.5);
299         solenoidfunc();
300         moveInY(2.5);
301         solenoidfunc();
302         moveInX(-2.5);  // moving back to start position in cell
303         moveInY(-2.5);  // moving back to start position in cell
304         break;
305     case 'e':
306         solenoidfunc();
307         moveInX(2.5);
308         solenoidfunc();
309         moveInX(-2.5);  // moving back to start position in cell
310         moveInY(2.6);
311         moveInY(2.5);
312         solenoidfunc();
313         moveInY(-2.6);  // moving back to start position in cell
314         moveInY(-2.5);  // moving back to start position in cell
315         break;
```

case 'q':
  solenoidfunc();
  moveInX(2.5);
  solenoidfunc();
  moveInY(2.5);
  solenoidfunc();
  moveInX(-2.5); // moving back to start position in cell
  moveInY(2.5);
  solenoidfunc();
  moveInY(-2.5); // moving back to start position in cell
  moveInY(-2.5); // moving back to start position in cell
  break;

case 'o':
  solenoidfunc();
  moveInX(2.5);
  moveInX(2.5);
  solenoidfunc();
  moveInX(-2.5); // moving back to start position in cell
  moveInY(2.5);
  solenoidfunc();
  moveInY(-2.5); // moving back to start position in cell
  moveInY(-2.5); // moving back to start position in cell
  break;

case 'p':
  solenoidfunc();
  moveInX(2.5);
  solenoidfunc();
  moveInX(-2.5); // moving back to start position in cell
  moveInY(2.5);
  solenoidfunc();
  moveInY(-2.5); // moving back to start position in cell
  moveInY(-2.5); // moving back to start position in cell
  break;

case 'g':
  solenoidfunc();
  moveInX(2.5);
  solenoidfunc();
  moveInY(2.5);
  solenoidfunc();
  moveInX(-2.5); // moving back to start position in cell
  solenoidfunc();
  moveInY(2.5);
  solenoidfunc();
  moveInY(-2.5); // moving back to start position in cell
  break;
APPENDIX A. ARDUINO CODE

```c
361  moveIN(-2.8);  // moving back to start position in cell
362  moveIN(-2.5);  // moving back to start position in cell
363  break;
364  case 'c':
365    solenoidfunc();
366    moveIN(2.5);
367    solenoidfunc();
368    moveIN(2.8);
369    solenoidfunc();
370    moveIN(-2.5);  // moving back to start position in cell
371    moveIN(2.5);
372    solenoidfunc();
373    moveIN(-2.5);  // moving back to start position in cell
374    moveIN(-2.5);  // moving back to start position in cell
375    break;
376  case 'd':
377    moveIN(2.5);
378    solenoidfunc();
379    moveIN(-2.5);  // moving back to start position in cell
380    moveIN(2.5);
381    solenoidfunc();
382    moveIN(2.8);
383    solenoidfunc();
384    moveIN(-2.5);  // moving back to start position in cell
385    moveIN(-2.8);  // moving back to start position in cell
386    break;
387  case 'e':
388    moveIN(2.8);
389    solenoidfunc();
390    moveIN(2.8);
391    solenoidfunc();
392    moveIN(-2.8);  // moving back to start position in cell
393    solenoidfunc();
394    moveIN(2.5);
395    solenoidfunc();
396    moveIN(-2.5);  // moving back to start position in cell
397    moveIN(-2.5);  // moving back to start position in cell
398    break;
399  case 'f':
400    solenoidfunc();
401    moveIN(2.5);
402    moveIN(2.8);
403    solenoidfunc();
404    moveIN(2.5);
405    solenoidfunc();
```

28
moveInX(-2.5); // moving back to start position in cell
moveInY(-2.5); // moving back to start position in cell
moveInY(-2.5); // moving back to start position in cell
break;
case 'o':
solenoidfunc();
moveInY(2.5);
solenoidfunc();
moveInY(2.5);
solenoidfunc();
moveInX(2.5);
solenoidfunc();
moveInX(-2.5); // moving back to start position in cell
moveInY(-2.5); // moving back to start position in cell
moveInY(-2.5); // moving back to start position in cell
break;
case 'p':
moveInX(2.5);
solenoidfunc();
moveInY(2.5);
solenoidfunc();
moveInY(2.5);
solenoidfunc();
moveInX(2.5); // moving back to start position in cell
moveInY(2.5); // moving back to start position in cell
moveInY(-2.5); // moving back to start position in cell
break;
case 'x':
solenoidfunc();
moveInX(2.5);
solenoidfunc();
moveInY(2.5);
moveInY(2.5);
solenoidfunc();
moveInX(-2.5); // moving back to start position in cell
solenoidfunc();
moveInY(-2.5); // moving back to start position in cell
moveInY(-2.5); // moving back to start position in cell
break;
case 'y':
solenoidfunc();
moveInX(2.5);
solenoidfunc();
moveInY(2.5);
APPENDIX A. ARDUINO CODE

```c
solenoidfunc();
movelX(2.5);
solenoidfunc();
movelX(-2.5);  // moving back to start position in cell
solenoidfunc();
movelX(-2.5);  // moving back to start position in cell
movelY(-2.5);  // moving back to start position in cell
break;
case 'z':
solenoidfunc();
movelX(2.8);
movelY(2.8);
solenoidfunc();
movelX(2.8);
solenoidfunc();
movelX(-2.6);  // moving back to start position in cell
solenoidfunc();
movelX(-2.6);  // moving back to start position in cell
movelY(-2.6);  // moving back to start position in cell
break;
/*
case 'A':
solenoidfunc();
movelX(2.8);
movelY(2.8);
movelX(2.8);
movelX(2.8);
solenoidfunc();
movelX(-2.6);  // moving back to start position in cell
movelY(-2.6);  // moving back to start position in cell
movelX(-2.6);  // moving back to start position in cell
break;
case 'g':
movelX(2.8);
solenoidfunc();
movelY(2.8);
solenoidfunc();
movelX(-2.6);  // moving back to start position in cell
solenoidfunc();
movelX(-2.6);  // moving back to start position in cell
movelY(-2.6);  // moving back to start position in cell
break;
case 'D':
movelX(2.8);
solenoidfunc();
movelY(2.8);
solenoidfunc();
movelX(2.8);
movelX(-2.6);  // moving back to start position in cell
solenoidfunc();
movelX(-2.6);  // moving back to start position in cell
movelY(-2.6);  // moving back to start position in cell
break;
case '8':
movelX(2.8);
```

30
moveY(2.6);
moveY(2.5);
solenoidfunc();
moveX(-2.0); // moving back to start position in cell
moveX(-2.5); // moving back to start position in cell
solenoidfunc();
moveY(-2.8); // moving back to start position in cell
break;

}

default: // if the input character is not matching any
cases it will do the default
Serial.println("Error, the letter does not exist in library");
moveX(6.0); // move to next letter
break;
}

}

void PrintNumber(int number)
{
  /* Printing numbers in braille is done by first doing the number sign and then printing
  the corresponding letter for each number. Where number 1 is a, 2 is b, 3 is c, 4 is d,
  5 is e, 6 is f, 7 is g, 8 is h, 9 is i and 0 is j.
   */
  moveX(4.0); //moved in X for a new number
  //first making the number sign, and then printing the corresponding letter for each number
  moveX(2.5);
solenoidfunc();
moveX(2.6);
solenoidfunc();
moveX(2.5);
solenoidfunc();
moveX(-2.6);
moveX(-2.8);
moveX(-2.6);

switch (number)
{
  // printing corresponding letter for number 1
  case '1':
    PrintLetter('a');
    break;
```cpp
// B

case '2':
    PrintLetter('b');
    break;

case '3':
    PrintLetter('c');
    break;

case '4':
    PrintLetter('d');
    break;

case '5':
    PrintLetter('e');
    break;

case '6':
    PrintLetter('f');
    break;

case '7':
    PrintLetter('g');
    break;

case '8':
    PrintLetter('h');
    break;

case '9':
    PrintLetter('i');
    break;

case '0':
    PrintLetter('j');
    break;

default:  // if the input character is not matching any cases it will
    // do the default
    Serial.println("Error, the number does not exist in library");
    moveInX(6.0);  // move to next letter
    break;

// prints punctuations such as comma, semicolon, exclamation mark etc
void PrintPunctuation(char punctuation) {
    switch (punctuation) {
        case '.':
            moveInY(3.5);
            break;
        // ... other cases
    }
}
```
moveInY(2.8);
solenoidfunc();
moveInY(-2.5);  // moving back to start position in cell
moveInY(-1.8);  // moving back to start position in cell
break;
case '3':
  moveInY(2.5);
solenoidfunc();
moveInY(2.8);
moveInX(0.6);
solenoidfunc();
moveInX(-1.6);  // moving back to start position in cell
moveInX(-2.5);  // moving back to start position in cell
break;
case '4':
  moveInY(0.5);
  moveInX(2.8);
solenoidfunc();
moveInX(-3.6);  // moving back to start position in cell
solenoidfunc();
moveInY(2.8);
solenoidfunc();
moveInY(-2.5);  // moving back to start position in cell
moveInY(-1.9);  // moving back to start position in cell
break;
case '5':
  moveInY(2.8);
solenoidfunc();
moveInX(0.6);
solenoidfunc();
moveInX(-2.6);  // moving back to start position in cell
moveInX(-3.6);  // moving back to start position in cell
break;
case '6':
  moveInY(2.8);
solenoidfunc();
moveInY(3.5);
solenoidfunc();
moveInY(-2.5);  // moving back to start position in cell

break;
case "'/':
    moveIn(2.6);
solenoidfunc();
    moveIn(-2.6);  // moving back to start position in cell
    moveIn(2.6);
    moveIn(2.5);
solenoidfunc();
    moveIn(-2.6);  // moving back to start position in cell
    moveIn(-2.6);  // moving back to start position in cell
    break;
case "":
    moveIn(2.6);
    moveIn(2.5);
solenoidfunc();
    moveIn(2.6);
solenoidfunc();
    moveIn(-2.6);  // moving back to start position in cell
    moveIn(-2.6);  // moving back to start position in cell
    moveIn(-4.0);  // move to next letter
    break;
default:    // if the input character is not matching
    // any cases it will do the default
    Serial.println("Error, the punctuation does not exist in library");
    moveIn(4.0);  // move to next letter
}

// will take input from serial monitor and then send each character to its printing function
void ReadInput()
{
    //Serial.println("wad will du prints");
    while(Serial.available()>0 && newData == false)      //don't read unless
    {
        if(index<100)
        {
            OneCharInput=Serial.read();          // reads in one character
            Serial.println(OneCharInput);        // checks if its not "/n" which means
            if (OneCharInput != EndMarker)      // end of input
            {
                indatastring[index]=OneCharInput;  // saves the character in a string with
                index++;                          // position as its index
                index++;                          // plus 1 for next character
// tells it that this is the end of the string, so it stops reading the subsequent byte of the memory that aren't actually part of the string.
newData = true;
// tells the function that it is completed with reading // in the message and now the new data can be printed
index = 0;
//Serial.println("last in");
break;
}
return indatastring;
}

for (int i=0; i < strlen(indatastring); i++) // looping for each character in the string
{
    Serial.println(indatastring[i]);
    if (stepsCounterX<592) // If it is no room for more characters in that row, 592 is how many steps the stepper motor (horizontal) can take
    {
        StepperX->step(867, BACKWARD, DOUBLE); // &67 because don't want it to hit the switcher
        moveIn(10.0); // for a new row
        stepsCounterX=15; // resets the stepscounter in X
    }

    //Serial.println(stepsCounterX);
    //Serial.println(stepsCounterY);

    if (stepsCounterY<466) // if it still is space vertically then it will print the character
    {
        // it will be sent to specific function depending on what it is
        if (isAlpha(indatastring[i]))
        {
            PrintLetter(indatastring[i]);
        }
        else if(isDigit(indatastring[i]))
        {
            PrintNumber(indatastring[i]);
        }
    }
}
APPENDIX A. ARDUINO CODE

```cpp
void ISR_Xswitch()
{
    // if horizontal switches is pressed down, this function will be run as a interrupt
    SwitchHFlag = 1;
    Serial.println("Xswitch pressed down");
}

void ISR_Yswitch()
{
    // if vertical switches is pressed down, this function will be run as a interrupt
    SwitchVFlag = 1;
    Serial.println("Yswitch pressed down");
}

void setup()
{
    // for solenoid
    pinMode(solenoid, OUTPUT);

    // for vertical motor (driven by A4988 motor driver, not shield)
    pinMode(stepPin, OUTPUT);
    pinMode(dirPin, OUTPUT);
    pinMode(sleep, OUTPUT);
}
```
```cpp
// for Serial monitor
Serial.begin(9600);
Serial.println("START 1");

// for interrupts with the switches, attachInterrupt(pin, function, triggers when pin low→high)
attachInterrupt(digitalPinToInterrupt(3), ISR_Xswitch, RISING); // RISING, triggers when the pin
// goes from low to high
attachInterrupt(digitalPinToInterrupt(2), ISR_Yswitch, RISING);

// for motor shield
AFMS.begin(); // create with the default frequency 1.6kHz
StepperX.setSpeed(20); // set the speed [rpm]

// homing in X-direction
SwitchXFlag=0;
while (SwitchXFlag<1) { // will stop when the horizontal limit switch is pressed down
  StepperX.backwardStep();
}

// homing in y-direction
SwitchYFlag=0;
while (SwitchYFlag<1) { // will stop when the vertical limit switch is pressed down
  StepperY.backwardStep();
}

// moves away from the switches before printing character
moveToX(2.5);
movetoY(2.5);
digitalWrite(sleep, LOW); // turning off the motor
Serial.println("What would you like to print?\n");

void loop()
{
  if (Serial.available() > 0)
  {
    ReadInput();
    newData = false;
    Serial.println("Will you print more?, I void loop");
  }
}
```
Bibliography


BIBLIOGRAPHY


[21] "Aluminium sheet — 500mm x 500mm x 3mm [Energy Class G]", Amazon, Available: //www.amazon.co.uk/Aluminium-sheet-500mm-3mm/dp/B00KRFA2AG. [20 May 2018]


