

POLITECNICO DI TORINO

Master of science in Mechatronics Engineering

Master thesis

Development of the low-cost solution for Braille display based on linear actuators with 3D printed mechanisms and servomotors.



**Politecnico
di Torino**



Supervisors:

Prof. Marcello Chiaberge

Prof. Sanjar Ruzimov

Author:

Zoirov Ulmas

Id (Polito): s254850

Id (TPUT): 04191

June 2021

Acknowledgements

This thesis work could not reach the pinpoint it is currently on without precious advice and practical help with the development and research of this project.

Marcello Chiaberge Want to thank for supervising my thesis work.

Sanjar Ruzimov supervised me from the very start until the very end, thoroughly leading me through all the process with temper and passion. Special thank for his enhanced language and strict comments and advises with no generalized. Non of the advises contradicted the previous one and were lined up in order and perfect logic.

Lazizbek Yusupov Has been from the very start of this project and used to be my perfect peer throughout all the processes of this project, from idea until the final product.

Sobirxonov Akbarxon motivated us to continue the work we stopped and decided not to return ever to continue and bring it to the form of ready product and still is making a great help in managing our team and obtaining new grants and searching investors.

Obidjon Ochilov assisted our project in developing, especially managing 3D printers and their settings - very tricky machines cumbersome to manage. He almost bleeds his fingers post cleaning the printed parts.

Shamsiyev Shaxobiddin assisted the code and hardware of the project. Special thanks for his passion in all the problems arose during the process of development.

Family was always there and still supporting even the least perspective ideas of our ill minds.

Abstract

Braille display is a display for the blind, but instead of LEDs it is composed of a matrix of dots that are mechanically operated with rising and falling capability. The matrix of one dot (or pin) consists of 8 dots, placed in 4x2 form. The combination of the dots with some risen and some in passive mode, i.e., fallen, gives us a chance to express all the possible ASCII symbols. There is a challenge among the scientific and production personnel to construct the cheapest and more reliable system for the blind to gain digital information and our team of engineers is a part of it and this thesis work is only a part of a bigger project carried out by.

This project is an engineering design and development project and is about development of a new affordable type of Braille Display. The many-fold cost reduction is achieved by replacing piezo-actuators with linearly driven mechanical systems and enhancing electronics. The mechanism is driven with cheap servo motors and is based on sliding with rising dots. The paper explains all the evolution of work performed, from the most primitive to the most optimal achieved till now with graphs and video links with description.

Table of Contents

1.Introduction.....	5
1.1 The reason of the project and what problems it is designated to solve.....	5
1.2 Brief History of development of Braille system and intro to the method.	6
1.3 The state-of-the-art of Braille Displays.....	7
1.4 The low-cost solution for Braille displays (solution short pathway) – the thesis objective.	11
2.Literature overview.....	13
2.0 Section description.....	13
2.1 Modular Low-Cost Braille Electronic Display.....	14
2.2 Application of the Latch mechanism for the Braille character.	15
2.3 The usage of servo motors as the Braille dot.....	16
2.4 Inductors with sliding mechanism.....	18
2.5 B.R.A.V.E. teamwork.	20
2.6 Circular sliding technology.	21
3. Development of the low-cost solution for Braille display.....	23
3.1 The solution using inductors - Slider 0.....	23
3.2 Solution using six linear actuators out of a servo motor- Slider I.	25
3.3 Solution using eight linear actuators out of a servo motor- Slider I. Slider II.	27
3.4 Solution using Stepper motors - Slider III.....	27
3.5 Solution using electromagnetic solenoid actuators - Slider IV & V.	30
3.6 Current solution using FDM 3d printers - Slider VI.....	31
4. Results.....	44
5. Discussion.....	46
6.Conclusion.....	48
7.References.....	50
8.Appendix.....	52
Appendix 1.....	52
Appendix 2.....	53
Appendix 3.....	54
Appendix 4.....	59
Appendix 5.....	60
Appendix 6.....	70

1.Introduction.

1.1 The reason of the project and what problems it is designated to solve.

There are 39 million blinds in the World according to WHO (latest statistics for 2010). They face various difficulties in their everyday life and one thing is the most significant in the whole picture[0].

Together with that nearly 90% of all of the visually impaired live in the third world countries [1] The blind of the whole world suffers from one thing, that makes their tedious lives even more difficult – the absence of access to the millions of terabytes of textual material available around on the internet or collection of intranets.

Most of the blind cannot use ordinary PCs of the general purpose and must gain information through special books specially designed for the blind. The books that are not printed but embossed to the special paper with tactile dots.

The dots are not placed like the ordinary alphabets in usual texts, but they represent special coding system, where each dot and the plane place, i.e., absence of dot stands for exact letter in alphabet (Appendix 1, Braille alphabet). The blind reads the book via touch sensing and perception happens through fingertips.

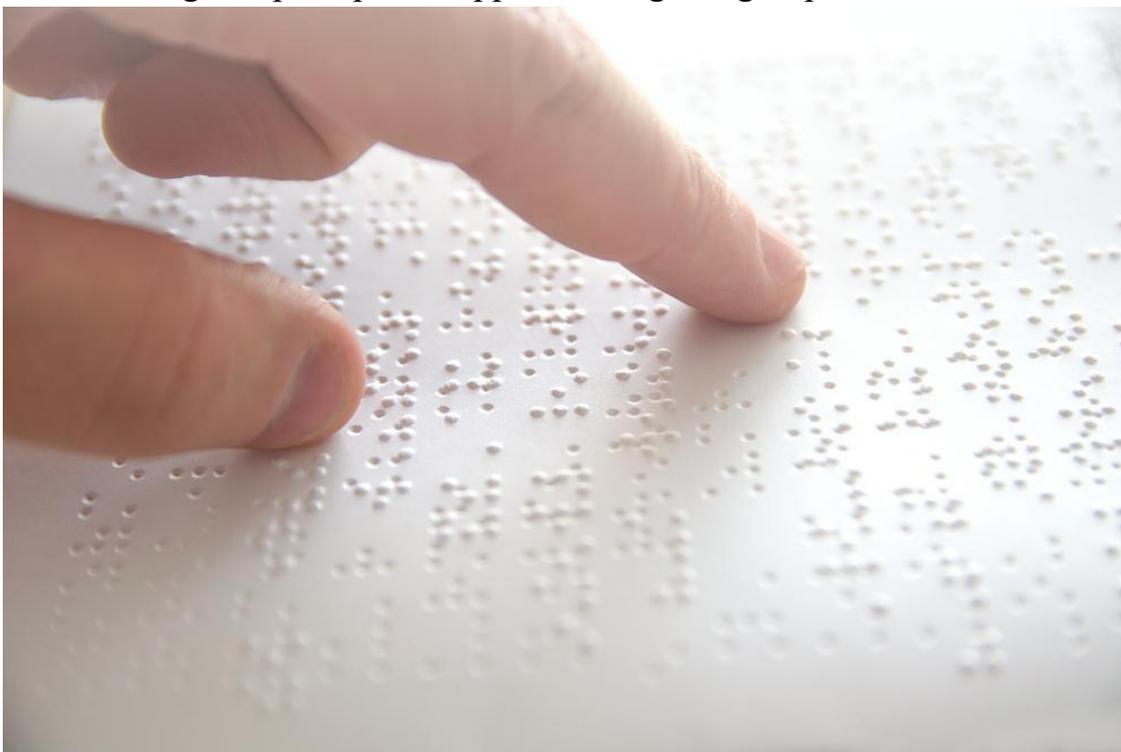


Figure 1. Braille paper [2].

Special paper is thicker than the one used in the offices (Figure 1) and consequently are more expensive, since different technology and requirements in quality standards rise the price exponentially. They have to buy, search this heavy and uncomfortable ‘bricks’ to have contact with textual world.



Figure 2. Braille book [3].

(Figure 2) gives an imagination about the size of braille books. Because of difficulty of manufacturing, the prices outnumber the prices of the ordinary books by several times [4]. The conversion of one ordinary book into Braille may cost up to 15000 USD and usual price for one embossed Braille book ranges from 200 all the way up to several thousand dollars.

1.2 Brief History of development of Braille system and intro to the method.

In the border of XVIII and XIX centuries the problem rises in the army of Napoleon Buonaparte, specifically in its part related to intelligence and secret police – in the battlefield it was difficult to read letters sent at night without exposing their position with the light of the candle[5]. The solution was to use the system of dots embossed on a thick paper. The technology was called Night Writing and there were trial in application of the technology in the battlefield, but

all of the trials were not successful. The writing method was presented by Charles Barbier and was further adopted for the blind children of specialized school for the blind in Paris. This methodology was shifted to the next level by the student at the school and reduced the alphabet of the Night Writing until the combination of six dots in a 3x2 matrix form and this enhancement is still bearing the name of the author (Appendix 1 – Braille alphabet).

Originally, the system of writing consisted of two instruments slate and stylus [6] where the paper is placed on a die with a grid of holes and the paper is pushed with a sharp pencil-shape part to emboss letters on the other side of the paper [7]. It was the earliest technology used by Louis Braille himself. The next generation of instruments is so called Perkins Braille [8a] that is a typewriter for the blind with only several keys, each of which represent one pin or dot of the Braille letter or the function of carrier return and line feed. Currently state-of-the-art electrical models are available, but still, it is not the best method of conveying the information. The next rise was a shift to completely different technology with no paper whatsoever with fully electronic controls – Braille displays.

1.3 The state-of-the-art of Braille Displays

There are various Braille Displays in the market [8b]. Braille Displays are tactile displays, i.e., they are read by the blind via touch sensing (Figure 3 and Figure 4) of the dots. All of the dots are controlled electrically and some of the dots rise, and others remain below the screen not to be sensed and that is the way literally any textual character is implemented for the blind.

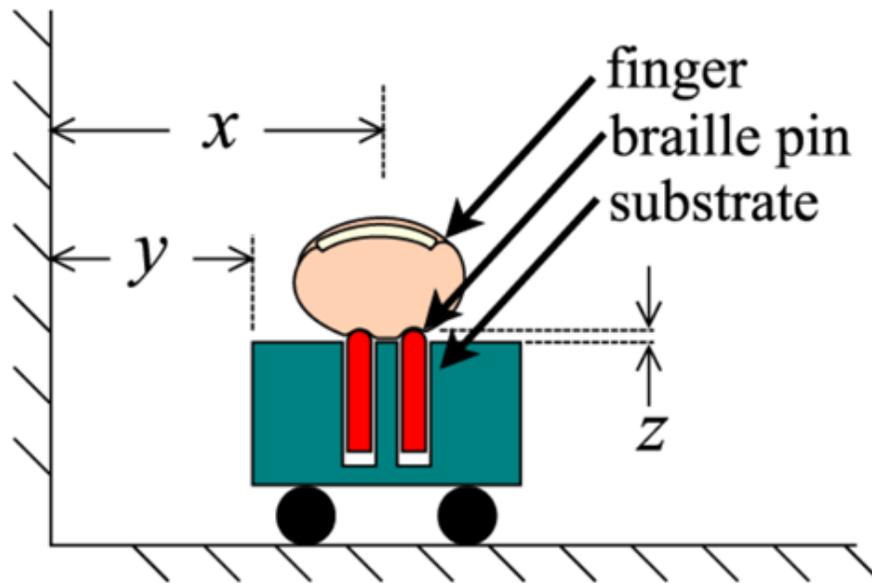


Figure 3. Finger sensing [\[9\]](#).

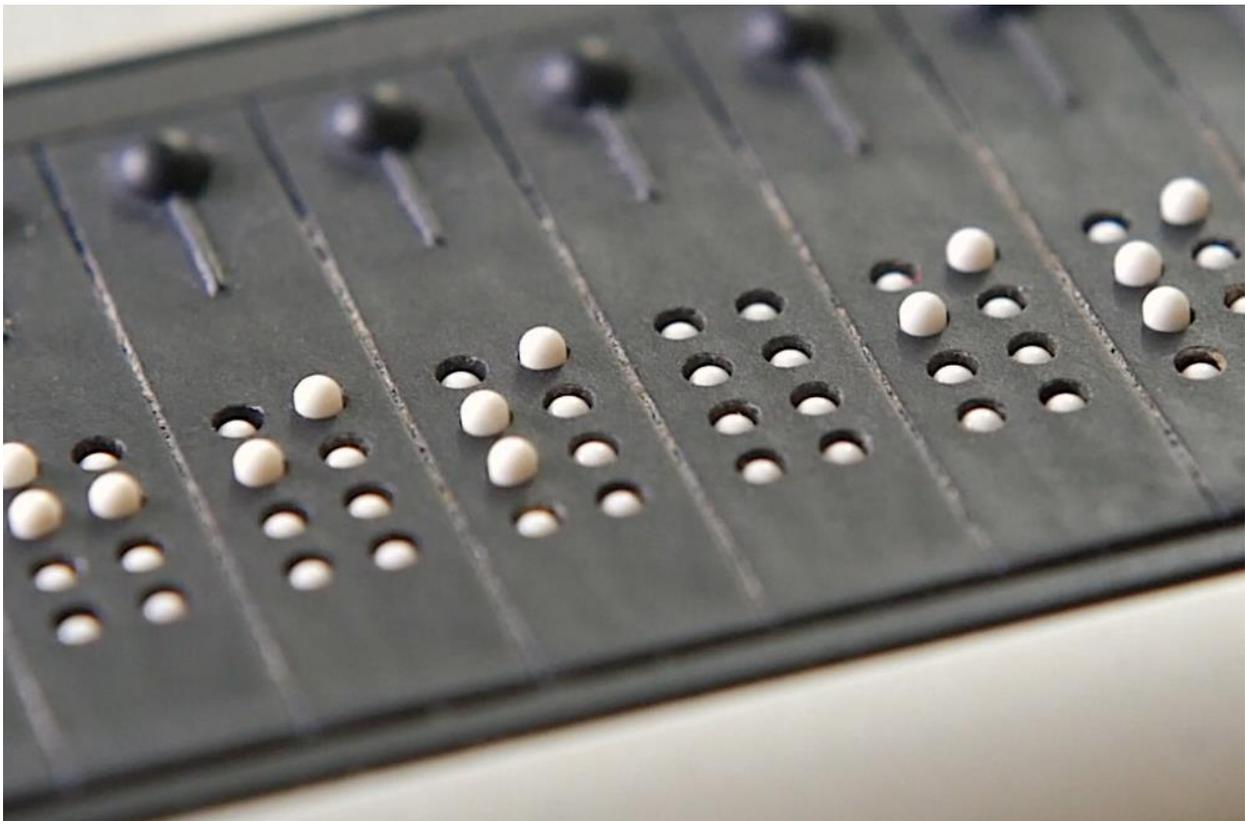


Figure 4. Dots in Braille display [\[10\]](#).

but all of them are with price tags beyond the limit of budget of most of the blind [11]. Therefore, there is need for decreasing the price of Braille Display by using much cheaper technologies for rising and returning the dots.



Figure 5. One of the Braille display in the market. [12].

The reason for high cost of this displays commercially available in the market lies behind the technology used to rise and lower those dots. The main reason for the high price is the technology used in manufacturing and design of the Braille Display in the market (Figure 5). The technology is used in actuators is called - piezo-actuators (Figure 6). They are manufactured using piezo crystals that are exceedingly rare on our planet.

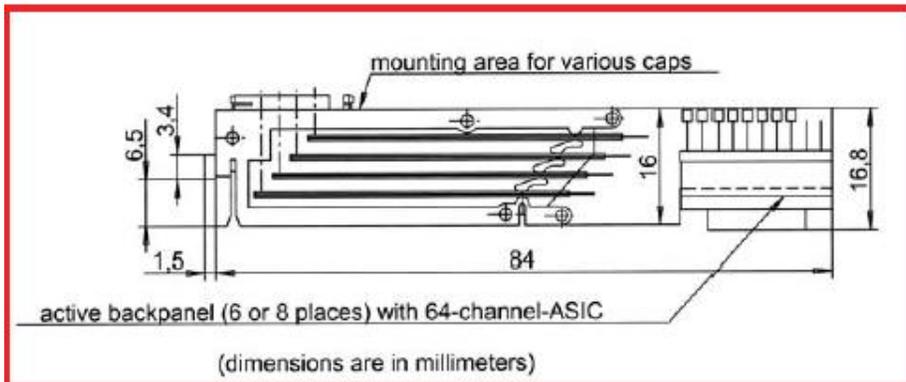
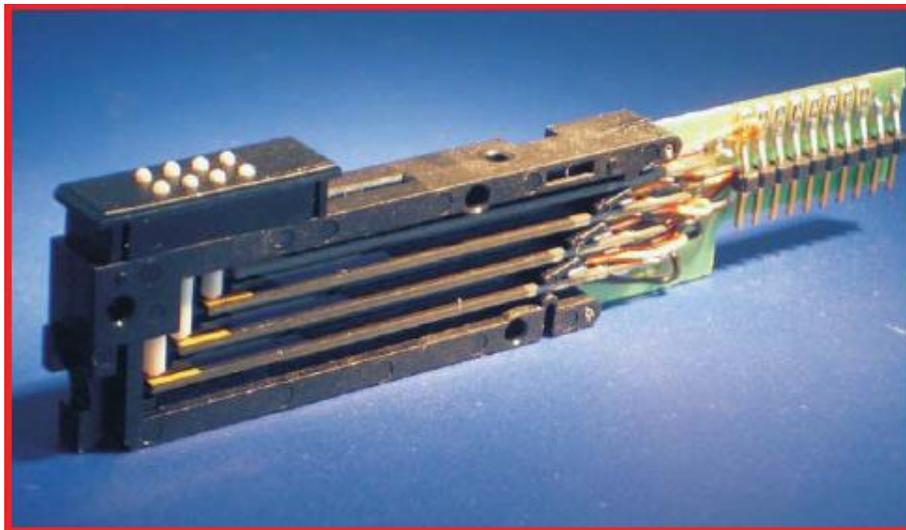


Figure 6. Piezo System in Braille display [13].

The main specifications of Piezo system [13].

- Dimensions (w x h x d) : 6.42 x 16.8 x 84 mm
- Dot spacing: 2.45 mm.
- Dot stroke: ca. 0.7 mm
- Cell spacing: 6.42 mm.
- Tactile force: minimum 0.17 Newton
- Connector: SIL 2.0 mm, 10 pins
- Drive electronic: Low power electronic on active back panel for 6 or 8 cells.

Those crystals have the ability to remember their shape, be they exposed to electricity. In Figure 6 one braille cell is depicted. The dots are controlled electrically via long sticks made out of piezo crystals and each time the electric current passes through them, they get bent up and move the dot up and with next

pulse of current the crystal is returned to previous position. Currently, all of the Braille displays available in the market are operated with the same technology and there is no other Braille display with employment of some other technology than that. The goal of this thesis work is to decrease the price of the Braille displays sharply by employing different technologies, that are of a lower price and fit them into Braille standards to develop the new generation of the displays.

1.4 The low-cost solution for Braille displays (solution short pathway) – the thesis objective.

The technology presented in the thesis about “Slider” series of affordable displays for the blind and currently the sixth structure is being developed to spread in the market. The main idea is to use cheap actuators like, tiny motors, stepper motors, actuators, and inductors to drastically reduce the price. “Slider VI” is based on using trick structure and cheap servomotors used in Arduino DIY projects, that are spread throughout the world and easily available almost everywhere. We are planning to develop a machine that could cost about 500 USD instead of 1200+ USD (Figure 7) currently available in the market.

6 Item(s)

		
Brailliant BI 40X braille display \$3,195.00	Brailliant BI 20X braille display \$1,895.00	Mantis Q40 \$2,495.00
		
Brailliant BI 14 braille display \$1,245.00	Brailliant BI 14 leather executive case \$95.00	Brailliant 80 braille display \$7,985.00

Figure 7. Braille display price range [14].

All the trials have been revealed in the further part of the work. And before the solution to move further to the next step, thorough research has been performed to continue the task improving the previous works even further.

“**The first step**” as it might seem obvious, was just replacing the piezo actuators with solenoids, and getting away with it. But it is not as simple as it may seem from the first look. Well, the first trial was with solenoids, since all the designs made with leverage of this technology seems persistent [15] but problems rise and the next move is taken to a simple system with servo motors.

“**The second step**” where for each pin one servo motor is bounded [16]. This system proves to be stable and durable, but it is too large and impossible to commercialize, since it does not fit Braille standards completely and it is almost impossible to build a display with at least 10 characters, since it becomes too large.

“**The third step**” then, it is explained the next move towards stepper motors and their direct use instead of servo motors [17] to diminish the space taken by each

character. But they prove to be unstable and even the smallest amount of dust made it inoperable.

“The fourth step” The next move is towards the combination of inductors with some tricky mechanisms, to overcome all the drawbacks of solenoids. It is planned to directly use the inductors for each pin individually, just increasing their stability [18] or employing a sliding mechanism, to move solenoids apart from center, to decrease the size of one character [19].

“The fifth step” currently, the project is in the phase of using sliding mechanisms with various motors [20] and with sliding bars, that are controlled with only two motors [21].

“The sixth step” Finally, the project comes up with a mechanism, where each pair of pins are controlled with one actuator [22] and dots are raised with straight bars moving back and forward, leveraged by rack and pinion [23].

2.Literature overview.

2.0 Section description.

This section explains other effective and most promising approaches in decreasing the price of the Braille Displays, both scientific papers and most promising works of startup developers throughout the Internet. Here the effort is made to hold the same order of resources, as it happened during developing of the affordable display. The whole project consists of several steps and during each one it was checked how is each idea able to be persistent enough to hold the positions in the market.

All of the researchers in the field might be divided into several groups in terms of methods of solving the problem. Some try to solve the problem through electromagnetic coils, while others push the direction of linear actuators, i.e., trying to solve the problem with super-expensive Braille displays via replacing piezo actuators with linear actuators driven by various motors.

The first group of researchers could be gathered as the party of electromagnets. They believe (it is not proven yet) that electromagnets are the best replacement for the dot actuator. Here is deeper explanation of each approach out of all, with the most chances in survival in the market.

2.1 Modular Low-Cost Braille Electronic Display.

The first one it was tried to replicate was Modular Low-Cost Braille Electronic Display (Figure 8) the prize winner in hackaday.io fair for the best startups [15]. The problems were faced with replication, since there was no SLA 3D printer in the whole country and FDMs could not do much, since they were designed specifically for high class precision SLA 3D printers. This is the very first drawback - it cannot be produced anywhere, including the developing countries, in R&D centers of which such printers are an excessively rare case. Another drawback is the strength of the pins - they do not stand against finger press. If the blind tries to read from that sign, the visually impaired must push on the character and there is no warranty, that they all will not fall below the sensible zone and become insensible.

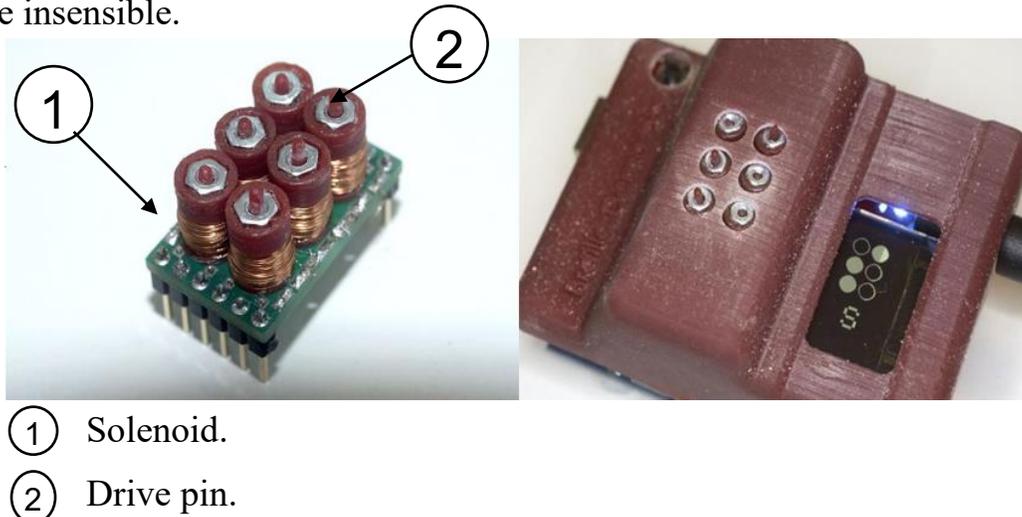


Figure 8. Modular Low-Cost Braille Electronic Display [15].

This method requires establishment of the whole inductor part production line and requires an extremely smooth ‘micromold’ technology to bring it up together. Yet, it is as fast as classical piezoelectric displays and completely fit the Braille standards. Anyway, not the last and the best solution.

It is visible in the figure 6 solenoids that drive 6 pins of the Braille display. As the current passes through the coil, it moves up the shaft with magnet up and it falls as the current flow stops. All of the solenoids are controlled though microcontroller, which controls current flowing through the coil and passes current through some pins, while cutting the circuit for others. That is the way the Braille letters are implemented from the Braille alphabet (Appendix 1, Braille alphabet).

2.2 Application of the Latch mechanism for the Braille character.

Team of researchers from South Korea and Germany proposed a mechanism with an inductor with a latch mechanism within. (Figure 9)

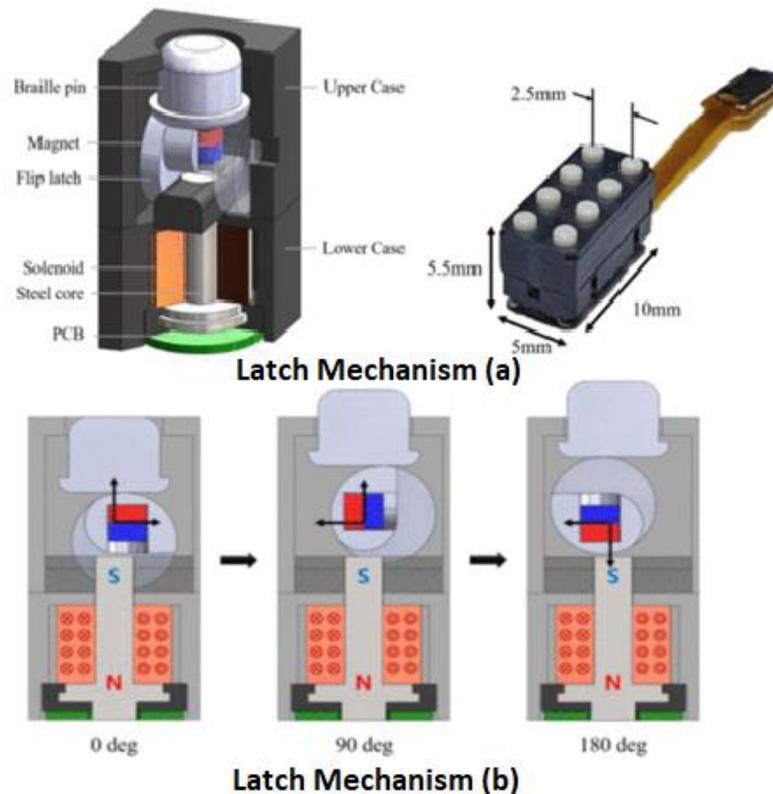


Figure 9. Latch Mechanism [19].

The working principle is the following – all of the mechanism consists of 8 parts. They all placed inside the case, that is for the purposes of the assembly process is divided into lower and upper parts. There is the Braille pin that is sensed directly by the user and to overall mechanism it connects via PCB connector.

The dot rises via flip latch that is connected directly to the magnet. It continues further below with an electromagnetic actuator, consisting of a coil and steel core.

In the passive state the Northern direction of magnet that is fastened on flip latch mechanism coincides with the polarity of the electromagnet. As the electromagnet switches its polarity, the magnet turns around to fit the polarity of the

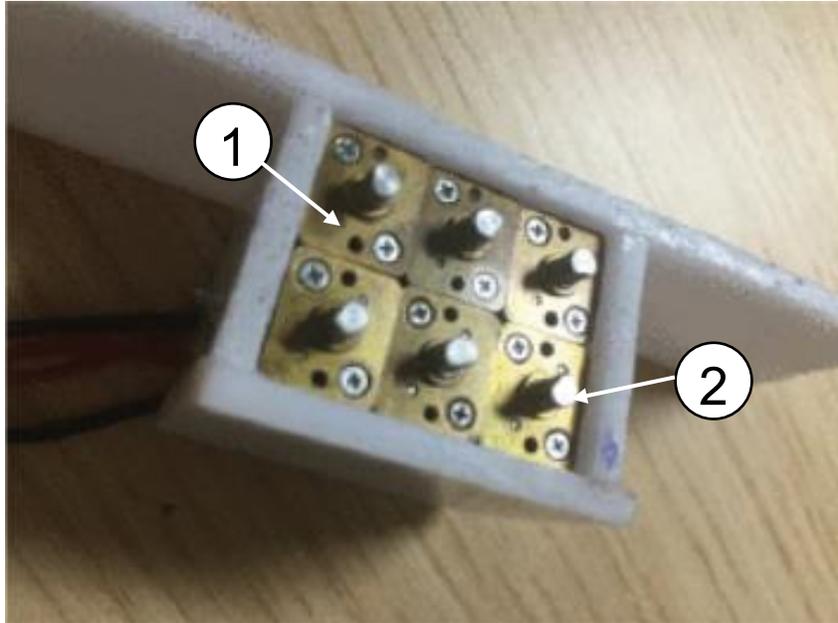
electromagnet and that is the case where flip latch fastened on the magnet rises the dot and not only rising, but it also holds the dot stable. So, the dot is stable to finger push and there is no risk of any dot not to be sensed or sensed improperly.

They are accurate and can be replicated either vertically or horizontally, giving a chance to high precision displays for the blind with the capability to display pictures and graphs. They are as fast as the classical ones [19]. But with that comes the problem, that is basically contradicting our idea - developing the display for the blind with a simple production process and low price. The idea proposed requires starting of a completely new production line with tiny parts.

2.3 The usage of servo motors as the Braille dot.

Some teams propose simple solutions, but there is little probability, that with the sizes of the actuators those researchers propose, somehow it might be possible to fit the pins to the Braille standards. Linear actuators (Figure 10) and the usage of servo motors directly for the pin (Figure 11) is one of the steps it took and have been lost on the track for quite a long time.

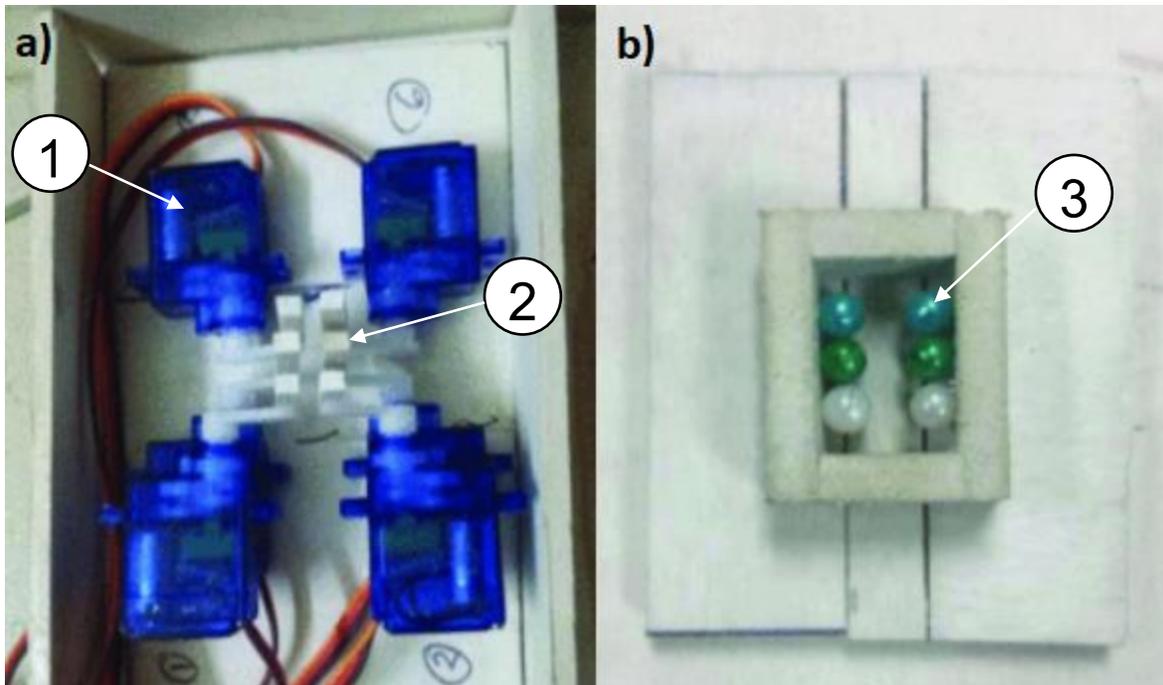
In Figure 10 There are 6 linear actuators and they are arranged in a form of a matrix in 3x2 size and that exactly fits Braille standards, but partially. These linear actuators are based on stepper motor with lead screw mechanism. As the controller sends the signal for stepper motor some of the linear actuators are actuated while others stay still and that is the way Braille letter for the Braille alphabet (Appendix 1, Braille alphabet) are implemented. They are united together with a housing in rectangular form and the tips of the linear actuators work like Braille pins.



- ① Stepper motor.
- ② Lead screw mechanism.

Figure 10. stepper motors with lead screw mechanism [\[17\]](#).

The work of the researchers depicted in (Figure 11 a and b) also use the principle of the linear actuator and they are arranged just like in the principle of the previous work. But the implementation is somewhat different. The system uses servomotors as the driving mechanism and pins placed on the shoulders of the servo motors move the balls that are employed like Braille pin in (Figure 11 b). Servo motors are also actuators by themselves, and they are used in various applications in robotics, automation. They are driven with PWM (pulse width modulation), where the angle of the servo motor is controlled by the width of the pulse. The range of the usual and the cheapest one is from 0 to 180 degrees and consists of a dc motor, encoder (resistive) and a reduction gear just before the contact interface.



- ① Servo motor.
- ② Balls.
- ③ Actuator.

Figure 11. Driving mechanism with servo motors [16].

2.4 Inductors with sliding mechanism.

But there is one research that could save this point of view [18], where inductors are not used directly, but in combination with sliding mechanism and stacking them from the top to a bottom (Figure 12), fits it within the standards but again it will be cumbersome to start the line for completely different design in the world of inductors.

The working principle is the following – as the inductive actuator gets the current, the iron core moves the direction shown in the (Figure 12a) (Power on). As the iron core moves it displaces the mechanism through R radius and e_1 part gets strict angle and Part F rises up. It has sufficient amount of strength to resist f_2 – the finger press power and f_2 is resisted by f_1 – the power of pull of the inductor. Each such mechanism is responsible for one Braille pin the are united into Braille cell

like shown in (Figure 12b). Only the length of Part F differs from pin to pin and the different lengths can be visible in (Figure 12c), where the pin in lower parts has longer shafts for the Braille pin, to get the even surface of the Braille cell.

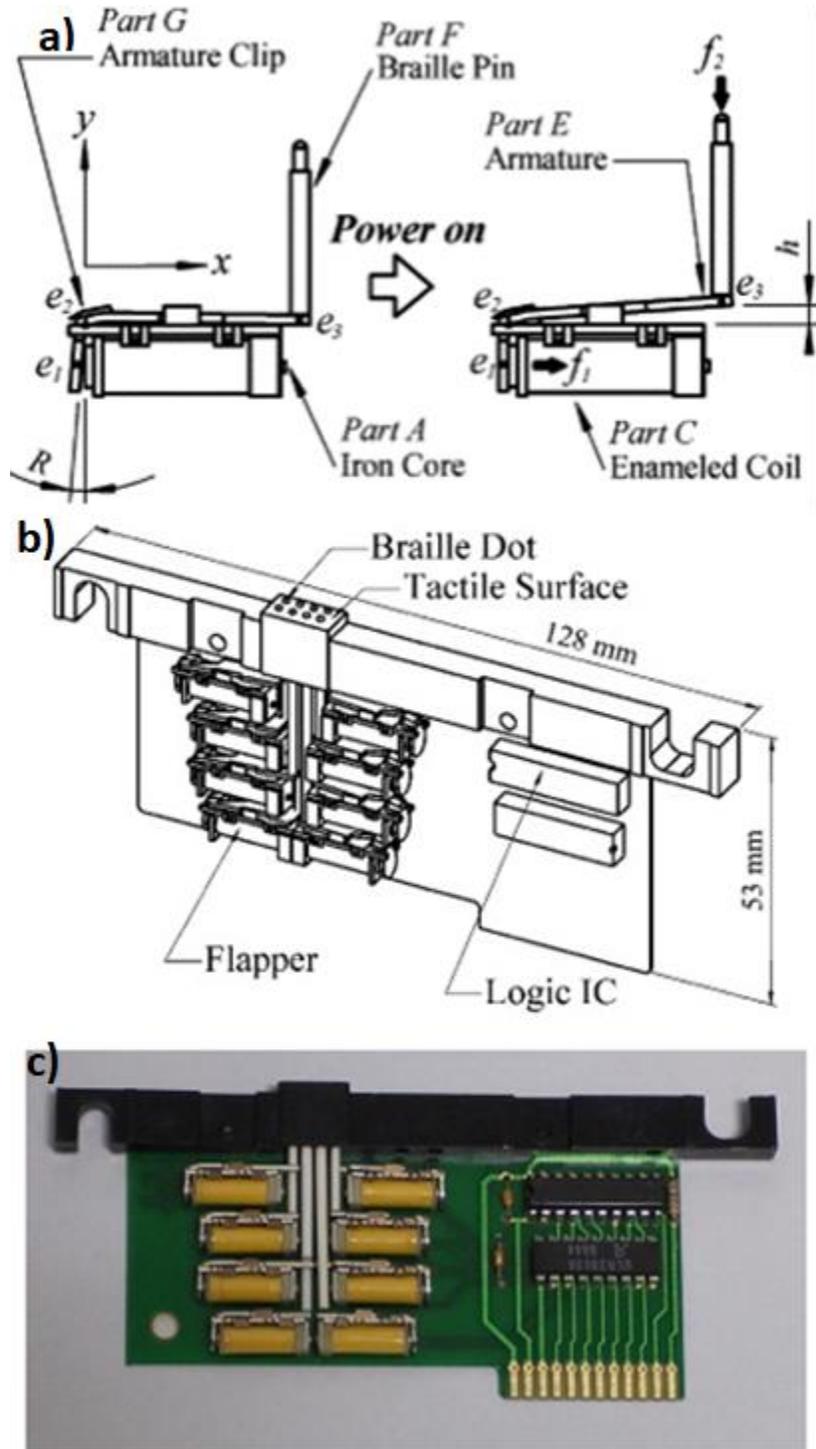


Figure 12. Combinational mechanism [18].

2.5 B.R.A.V.E. teamwork.

The most promising solutions, as it seems from stability of the system and the best competitors of this thesis work are the researchers proposing the idea of usage of cheap circular motors converted to a linear actuator and with added sliding mechanism to rise and retreat the pins. (Figure 13) The one presented in this work has been inspired by, is B.R.A.V.E. team with their profile in thingiverse platform with thousands of drawings and files that could be directly loaded to 3D printer [23]. But the development has back sides like, again employing of SLA 3D printers, not thoroughly thought mechanism to return the pins down and weak architecture of linear actuator. But the only idea of using a linearly straight bar to raise dots is worth working on.

But this project did not meet requirements to be followed for the following reasons:

The absence of technology or to high price of it and rareness made it almost impossible to follow up the project.

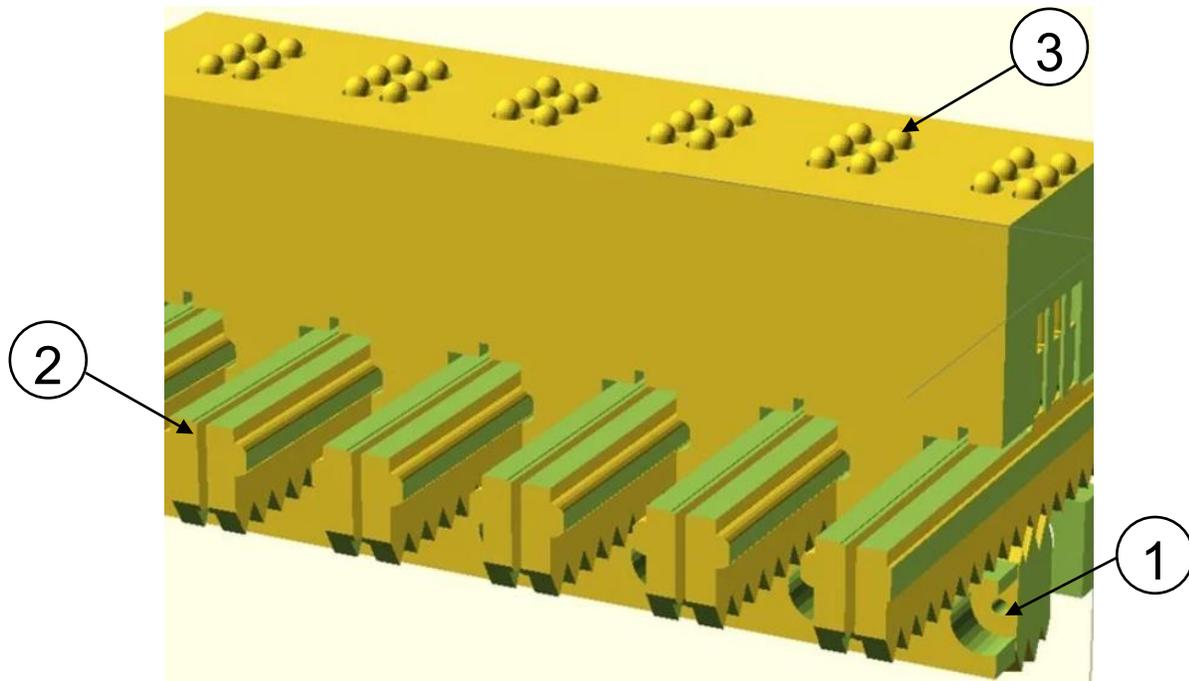
This project requires serious improvements and currently the work is being performed on:

Mechanism is impractical in ordinary FDM 3D printers, since parts are too tiny. The problem with control has not been solved - only DC motor has been connected, that obviously has no feedback to inform the controller about rotations etc.

No thought up the problem of putting the dots back into base. This problem is planned to be solved with springs, but it is cumbersome to produce such strings and difficult to find in the market. That requires customization with factory R&D centers, that adds more value to the price. Both researcher teams mentioned above, tried to solve these problems with SLA 3D printers, that are too expensive, rare, and cumbersome to manage.

Most of the researchers try to solve the problem with seemingly easy inductor coils to move neodymium magnets, but due reasons mentioned above (special R&D is required), this solution is not the best to follow up with.

Another group tries to solve it with some tricky mechanisms or at least mechanisms printable with relatively expensive 3D printers and materials (SLA).

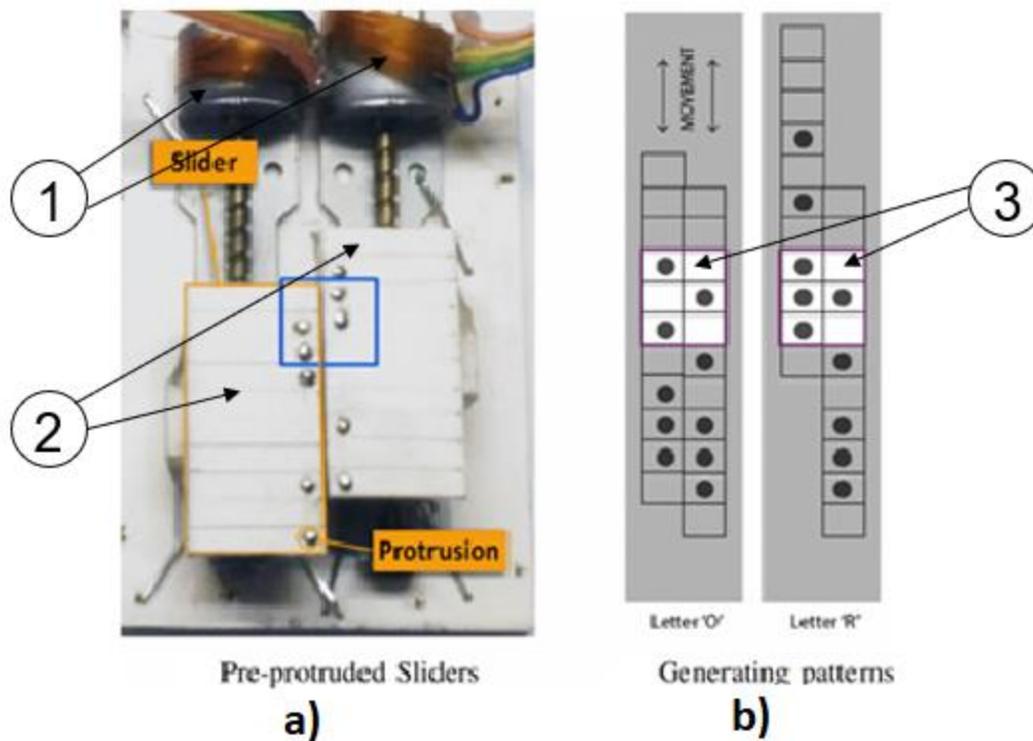


- ① Circular motor place.
- ② Sliding mechanism.
- ③ Pin.

Figure 13. B.R.A.V.E. Teamwork [23].

2.6 Circular sliding technology.

There is a more practical solution with the pins moving together with the bar (Figure 14), losing a chance to diminish in size further [21]. The working principle of the mechanism bases on two biggest and major parts – two linear actuators and two bars with pins on them. As the motors circulate clockwise and counterclockwise, two consecutive bars move forward and back, and the most important part of this process may be understood looking at Figure 14a. The two bars have an arrangement of dots on them, and they are moving back and forward bring the Braille letters in various positions. Figure 14b shows letters ‘o’ and letter ‘r’. This is the system of only one Braille cell and several cells must be placed in a row to get one display.



- ① Two linear actuators.
- ② Bars.
- ③ Braille cells.

Figure 14. mechanisms with servo and stepper motors[\[21\]](#).

The biggest drawback of this approach is too large size of the mechanism that cannot be decreased in size until Braille standards (Appendix 2, Braille standards). Another problem with this approach is its interface that is not comfortable for the visually impaired to use, since pins always move back and forward, it might bring into misunderstanding of the texts.

3. Development of the low-cost solution for Braille display.

3.1 The solution using inductors - Slider 0.

In this thesis trial has been made to replicate the previous designs in the field and drawbacks has been learned thoroughly and solving has started. The development of the project started from the most obvious - solenoid-to-pin mechanism. For that, the researcher tried to replicate the work done by MOLBED team to see if their model fits the requirements of the market. But the parts shown in the (Figure 15a) could not be printed on FDM 3D printer due to the low precision of the technology and hence the parts developed for SLA technology could not be widespread (Figure 15b), since the technology is relatively rarely met in R&D centers around the world and too high price of the printer itself and its material [\[24\]](#).

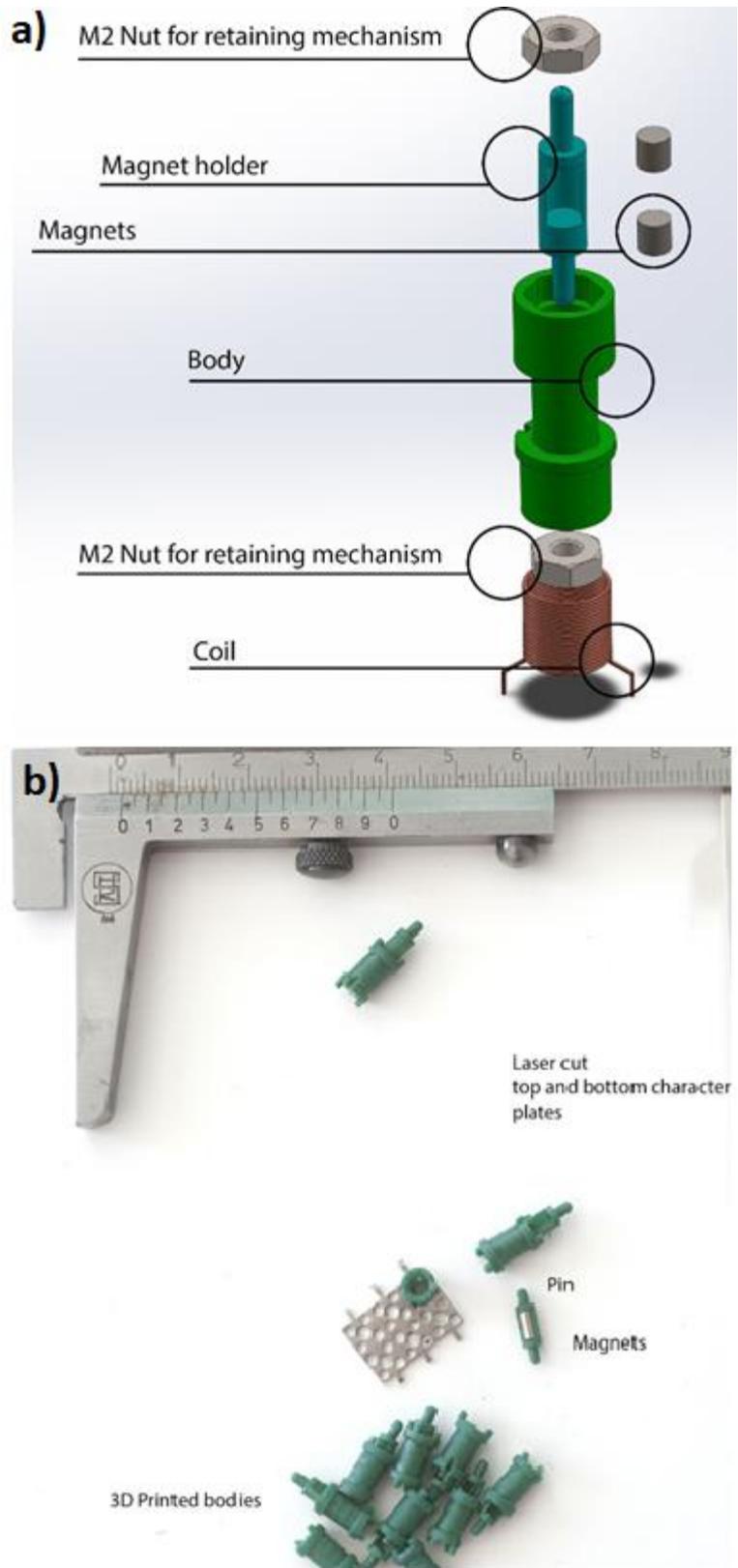


Figure 15. 3D parts of Modular Low-cost Braille Electronic Display [\[15\]](#).

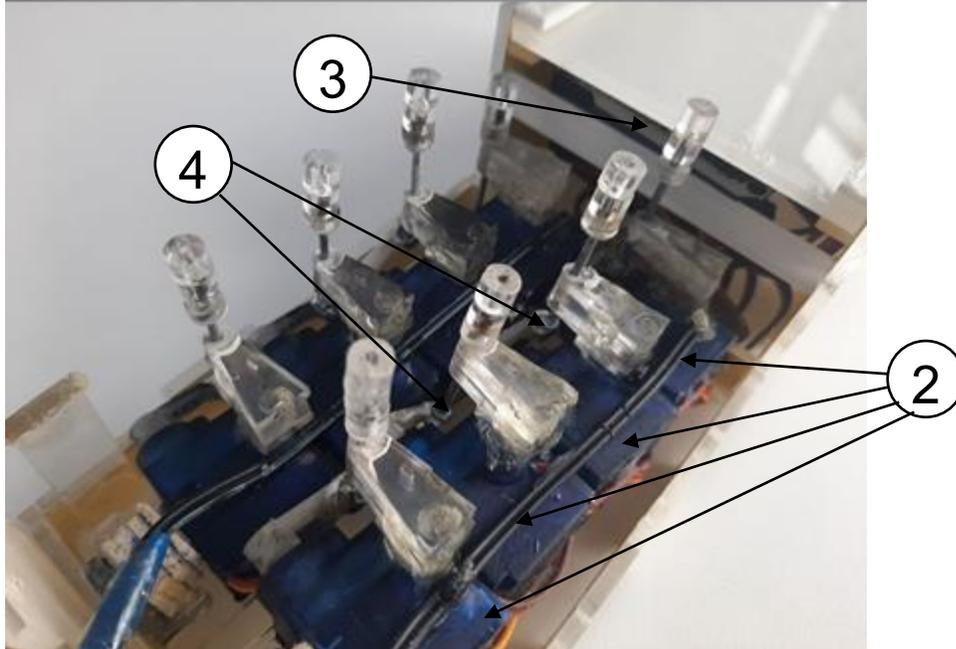
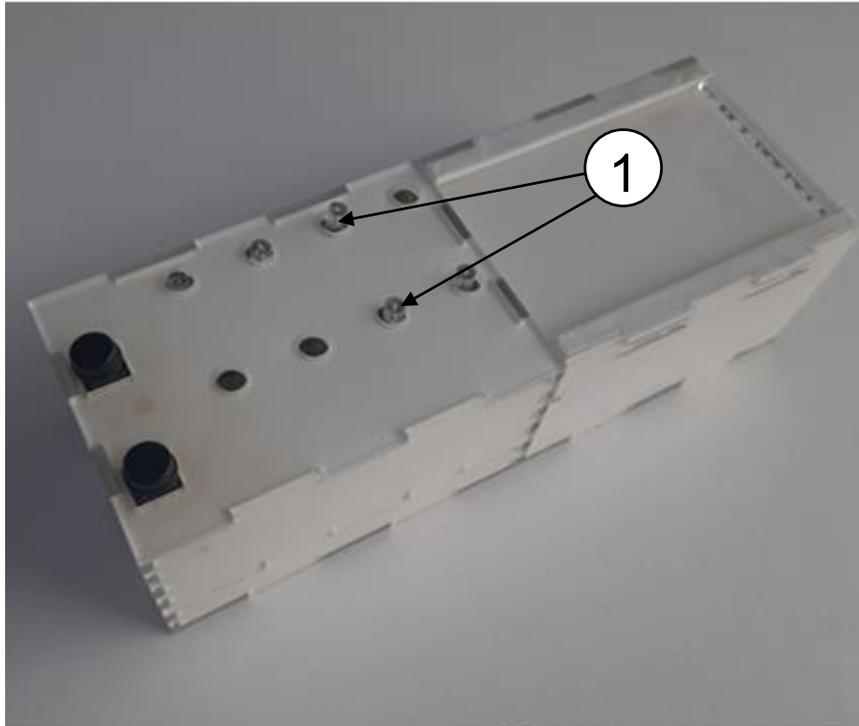
The coil is curved around the body and magnets are placed inside magnet holder and further the magnet holder itself is placed inside the body with a coil around it. Nuts are placed in both sides to stop the magnet holder from pulling out completely. As the coil is powered with a current and changing its polarity the magnet holder moves up and down and the pin part on magnet holder displaces up and down hiding completely, implementing on pin of the Braille cell. The current, as usually, is controlled via current controller that is controlled by a microcontroller and having moved exact pins it is possible to get Braille letter from the Braille alphabet (Appendix 1, Braille alphabet).

All has been thoroughly shown in the instructions [15] and could be easily replicated, from the size of magnets to get to the number of loops in the coil. Despite being unrealizable for most engineers, it has other weaknesses. The holding of the pin would require uninterrupted supply of DC for the whole period of holding the pin active (i.e., in raised state) and even in active state with finger push, it returns to passive state, making it difficult to recognize the sign.

3.2 Solution using six linear actuators out of a servo motor- Slider I.

After witnessing that it is impossible to build anything tangible with an inductor alone, decision has been made to move further with another solution and build the linear actuator out of a servo motor that looks very much like the work done by BRAC University researchers [16] (Figure 16. second prototype). The very first prototype was with only 6 dots and made of unstable and weak materials, but the next version was drawn with CAD and developed taking into account materials that are cut in laser mill.

As it might be seen from Figure 16, there are 8 linear actuators driven by the servo motor and all of them are together controlled by Arduino Mega controller (Arduino developed code in Appendix 3). As the servo motor performs circular movement, the dot rises, sliding on a tiny neodium magnet and as it circulates backwards, the pin falls inside the case, disappearing and that is the way Braille letters are implemented.



- ① Linear actuators.
- ② Servo motor.
- ③ Arduino Mega controller.
- ④ Neodymium magnet.

Figure 16. second prototype

3.3 Solution using eight linear actuators out of a servo motor- Slider I. Slider II.

It has proven to be relatively persistent construction with long-lasting mechanical parts, but was way too large, then it must be in accordance with the Standards and power-ineffective in all means, requiring too much of the energy at once and in general.

The prototype shown in Figure 16 is relatively more refined version of the previous one. Here the number of actuators has been increased to 8, to fit ASCII standards (Appendix 4, ASCII). With these 8 pins, any combination of which may be performed, it is possible to replicate any letter and sign existing today. The two buttons on the housing moves the cell through the string from letter to letter. Another improvement, it is possible to implement capital letter with two lowest pins in the matrix of dots.

3.4 Solution using Stepper motors - Slider III.

The next decision was to replace servo motors based linear actuators (Figure 16. b) with the tiniest linear actuators available in the market the linear actuators used in photo cameras to move the focus lenses and we designed linear actuators mechanism by 6mm linear screw stepper motors (Figure 16. a)

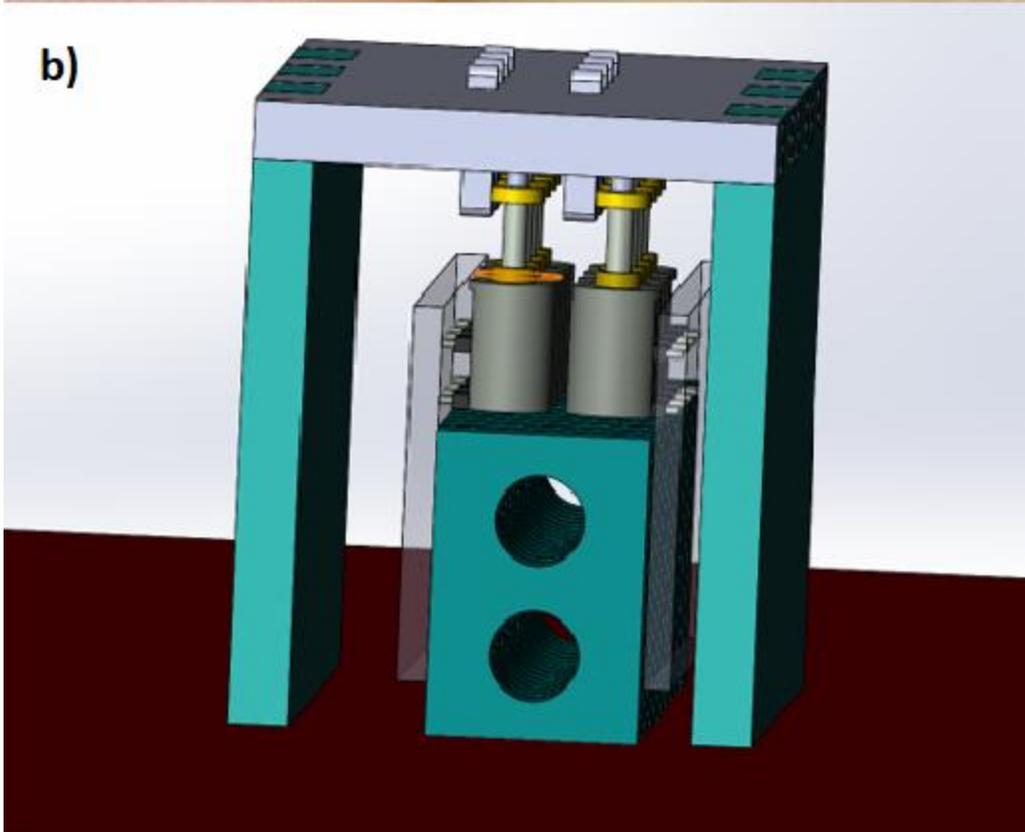


Figure 16. Braille display design with stepper motor.

They are placed each under one pin and were meant to raise the dot with the given signal from the controller. Theoretically, it had to do something, but practically all of the drawbacks made it the version with no rights to be invested further - after getting dusted for a little bit during a pair of hours, nuts got stuck on the screw and additional cleaning had to be performed in order to renew the machine working, the power of the motors were insufficient to overcome the friction between moving parts and super high definition (Figure 17) and precision was required to make it sustainable and hence too high price. It works for the same purpose as are all of the other versions and approaches – to implement Braille cell and fit its standards.

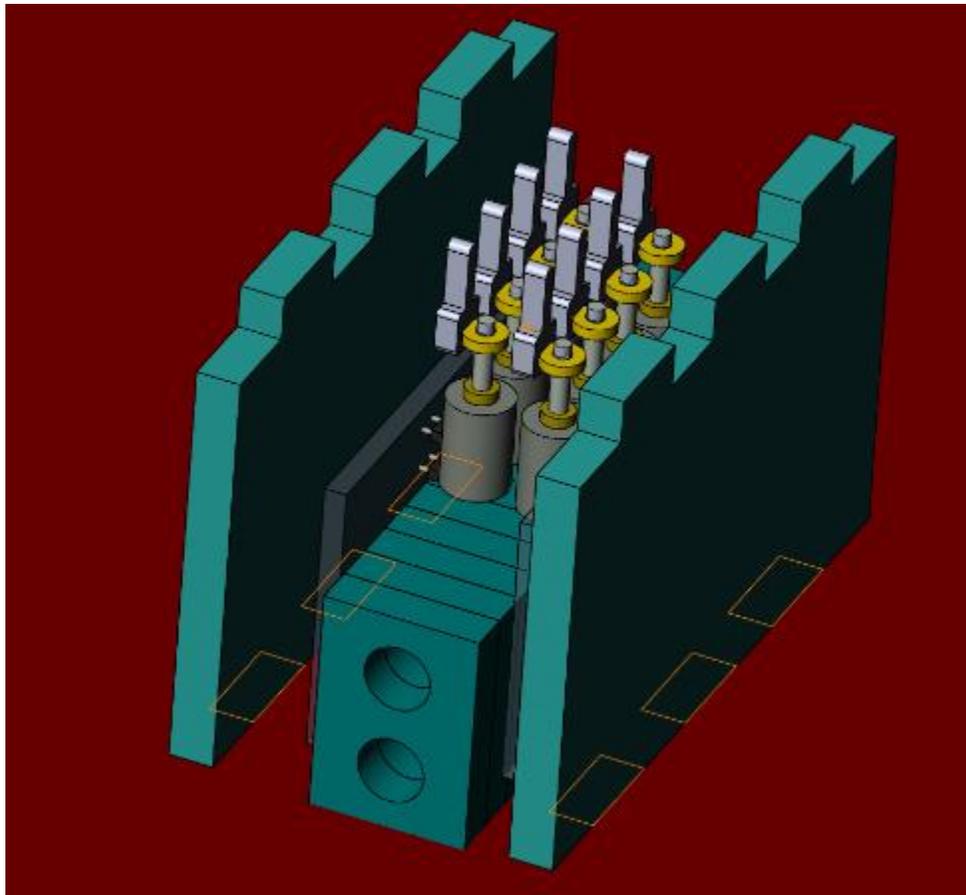


Figure 17. stepper motor view

3.5 Solution using electromagnetic solenoid actuators - Slider IV & V.

Eventually, it became possible to find the proper electromagnetic solenoid actuator in the market that has not even had to be engineered further and the production process could start immediately with just purchasing the proper product (Figure 18). It consists of a coil wrapped with plastic sheet, a magnet that returns back the shaft when it pulls up. There are comfortable fastening parts, that could be added to literally any mechanism and are controller through two wire – ground and VCC. As the machine is powered it moves either forward or backwards.

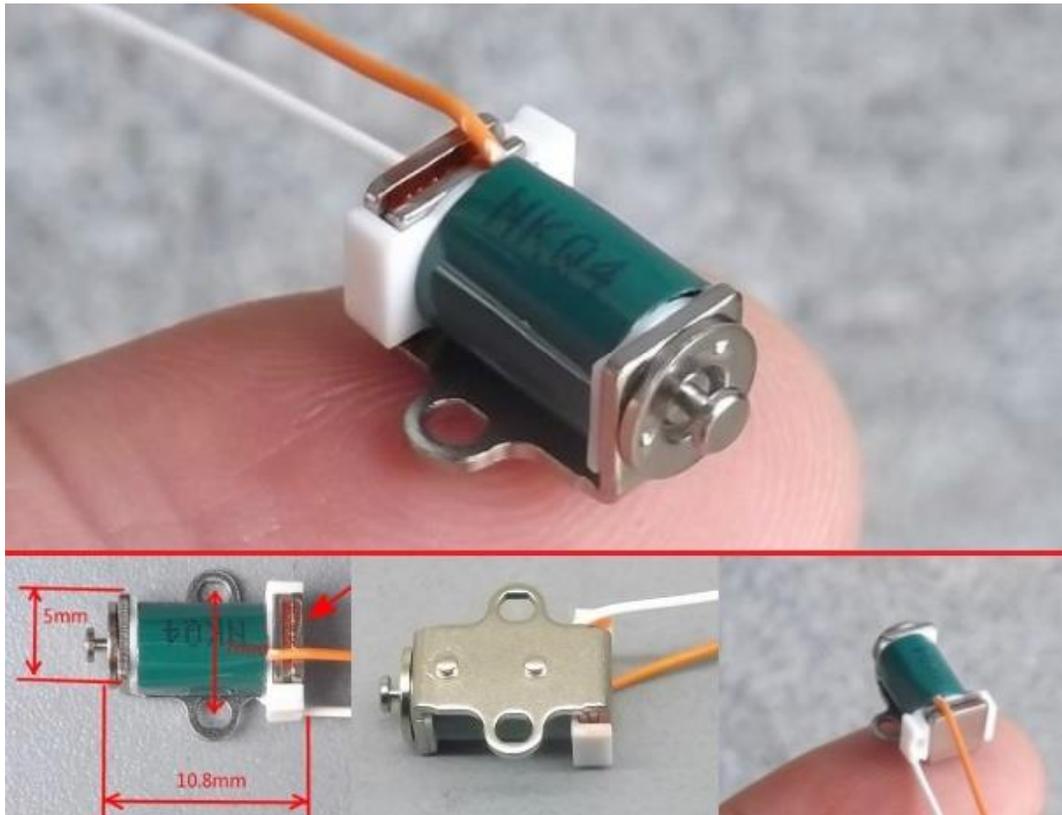


Figure 18. DC Electromagnetic Solenoid actuator [25].

The idea is to use a conversion mechanism that could convert horizontal movement into vertical with motors stacked one on another (Figure 19) to save the space and hold on with the Standards.

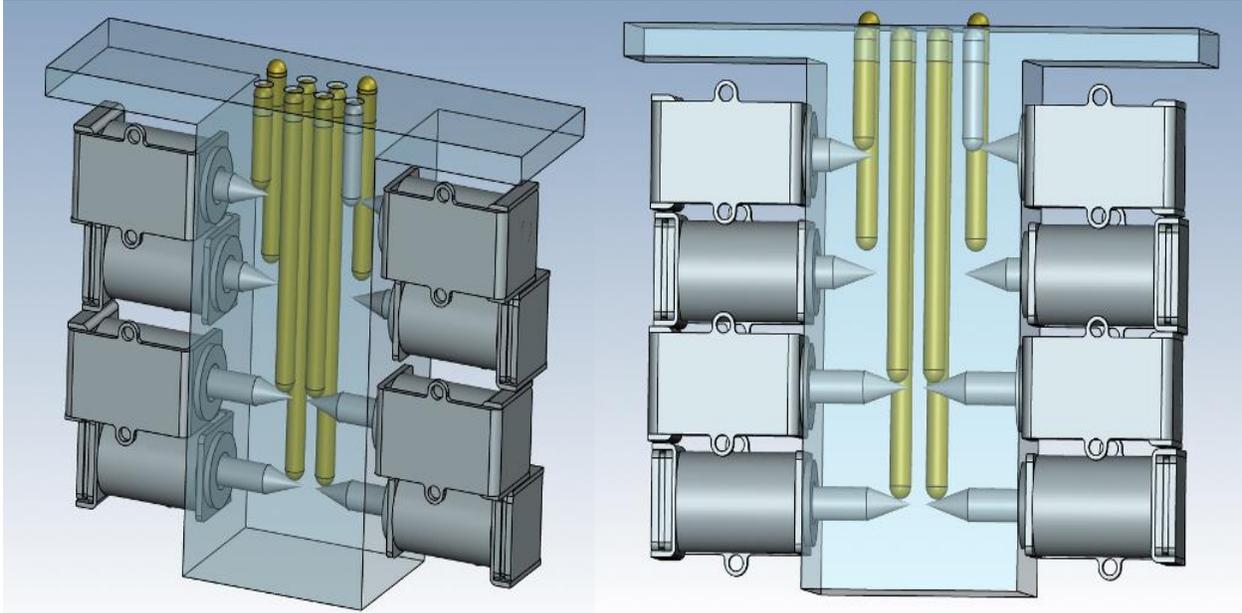


Figure 19. Conversion mechanism

Each electromagnetic actuator is placed in vertical stack, in order to save the space and be as close as possible to the Braille cell standards (Appendix 2, Braille standards). The shaft is sharpened in 30-degree inclined plane manner and when the shaft moves out, it moves the pin rod up and vice versa when pulled back. All of the electromagnetic actuators and pin rods for Braille cell.

But in the very stage of calculations this idea was dropped, because it became possible something that could radically decrease the time of the research and could help to overcome all of the drawbacks of all of the previous ideas and prototypes.

3.6 Current solution using FDM 3d printers - Slider VI.

In the faraway spot of the Internet, it has been encountered interesting designs with sliding bars, but individual pins [\[23\]](#) that rise with an inclined surface and return to its passive position with a spring and with DC motor converted to linear actuator. The idea had weak implementation and it was decided to move this idea forward,

first trying to replicate it with existing machines and capabilities of the R&D center the research was held.

In the project various mechanisms were reverse engineered, to apply them in 3D printing, since there are no theoretical methods to analyze the FDM 3D prints, due to their complexity. Relying on laws of Machine Design and Structural Mechanics, followed with trial and error in practically printed 3D prints.

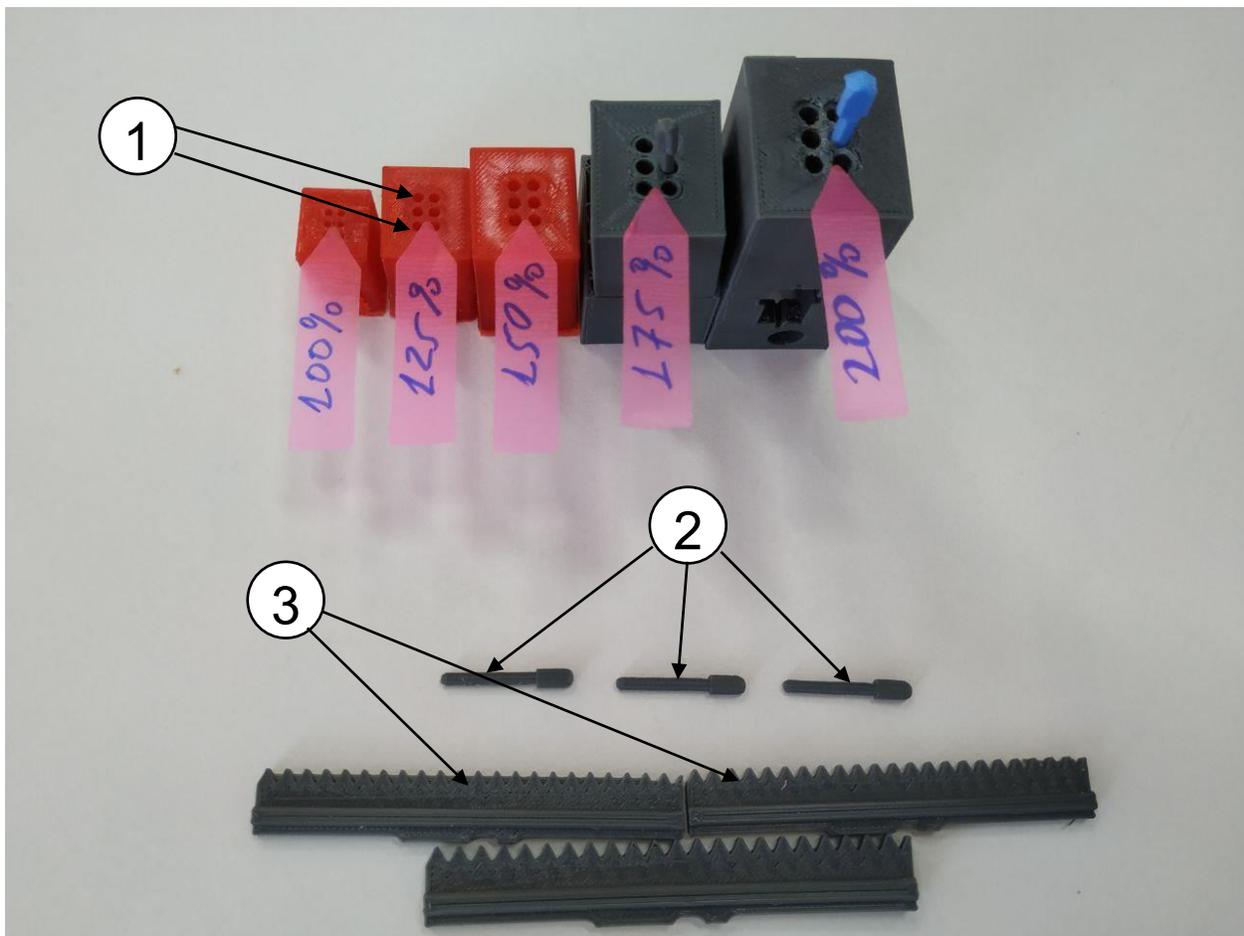
Since, in this case it was impossible to employ simulation tools and theoretical calculations were useless since there are no ones for 3D printed structures. Mostly an empirical approach was employed, to get the best outcome.

In this case, piles of 3D prints were made and even after printing they were measured with a caliper, because the accuracy of the ordinary FDM 3D prints is 0.5 mms. To decide the sizes of parts to use, the row of the parts was printed with 0.25 mm gap from the least possible and the largest possible, to get the most accurate mean value within the scope of all the printed parts. Iterations were made, when one period was chosen, other prints on various materials and models of 3D printers were performed, to get the universal value, might be printed in any part of the world, since the idea is to make calculations, that could be produced by literally any FDM 3D printer in any region.

(Figures 20) - All the process of learning this technology started from its replication and the first result shown, that this technology is impossible to replicate with FDM 3D printers and trial has been made, to bring its size into those, where it could be printed with the quality to operate mechanically. And eventually, 200% zooming has been chosen, to reach plausible strength of parts with no breaking with just connecting. It has appeared, that for 3mm diameter pin is able to hold the weight of around 100 grams horizontally and able to hold maximum press power of human.

(Figure 20) depicts all the prints that were examined for sufficient strength and capability of production. They were printed with various scales of 100% (i.e., original), 125% and all the way to 200% with 25% gap in-between. The 100% version proved to be with too weak pins and too small holes, and the level of accuracy was completely unacceptable for the proper operation. The experiment

and tests with connection has been observed, that 200% is the best match, where the parts are accurate enough to slide smoothly enough and the parts are thin enough, but not too thin to break right after taking in the hand. The holes for pins vary from trial to trial and in some point, it was impossible to operate properly to get the sustainable result. The experiment has started with six dot arrangement, since it is the simplest one to replicate and the material used was PLA (Polylactic Acid). The rows of prints were accurately registered and tagged with order numbers. In Figure 18 the process of checking for compatibility of the main block with a rack has been tested for both strength and accuracy.

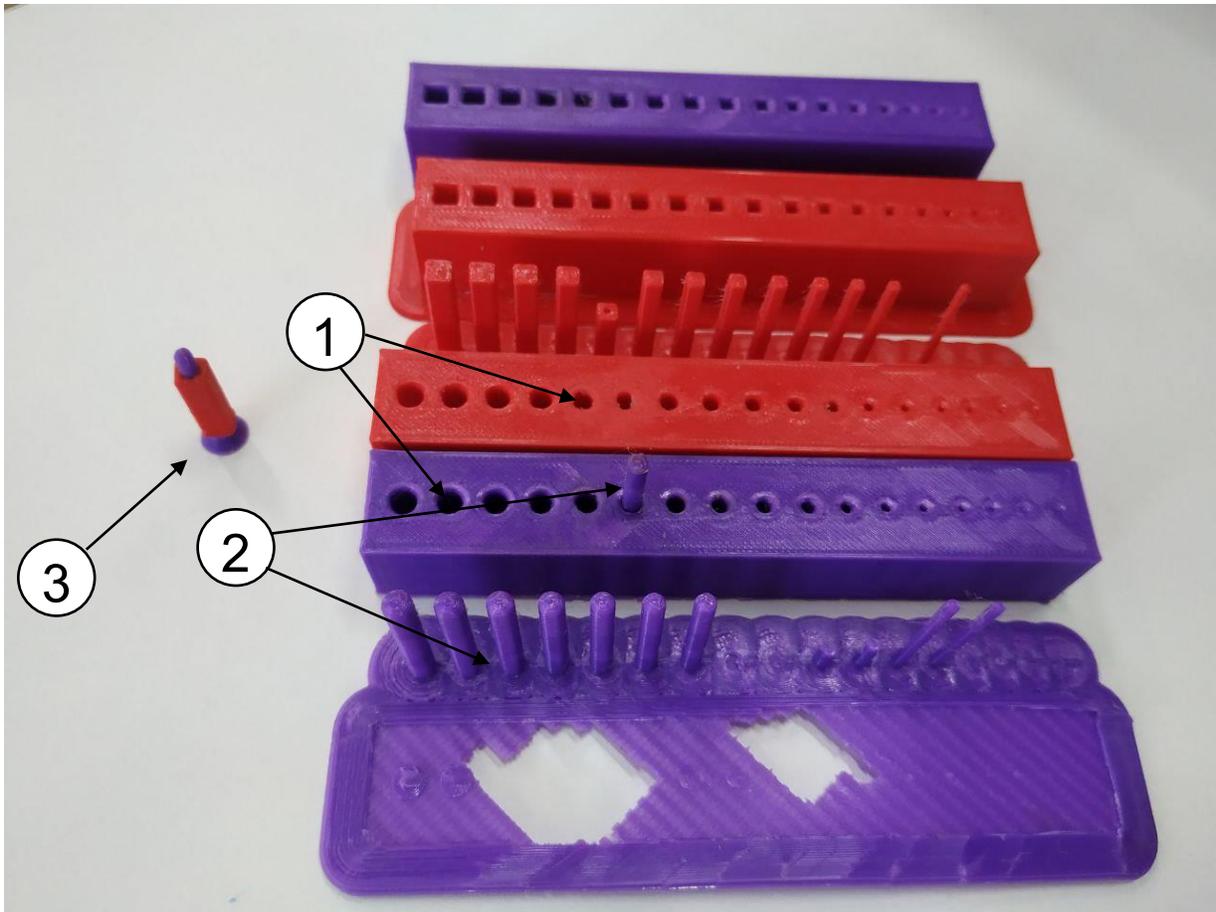


- ① Holes.
- ② Pins.
- ③ Racks.

Figure 20. Testing holes, pins, racks, and scale varying from 100 to 200%

Figure 20 shows the ones that passed to the next stage for further testing. The racks in this stage were strong enough for operating properly and the main blocks had the cylindrical holes with highest possible accuracy for the pins to move inside that bar. The sliding part of the bars were precise enough for the pins to slide accurately implementing the Braille cells.

(Figures 21)- It was important to choose the right relation between the hole and the pin. Since the precision of the mean FDM 3D printer is 0.25 mm, but anyway, it was not an exact result and whatever has been assigned in the instruction book, was not the most accurate data, and special printing and comparison had to be performed to check the best relationship between the hole and the pin. The experiments combining all the pins with holes has shown, that for average price 3D printer, best configuration for pin and hole size relationship is 0.25 mm of offset, to leave space for moving parts and lubrication of the mechanisms.

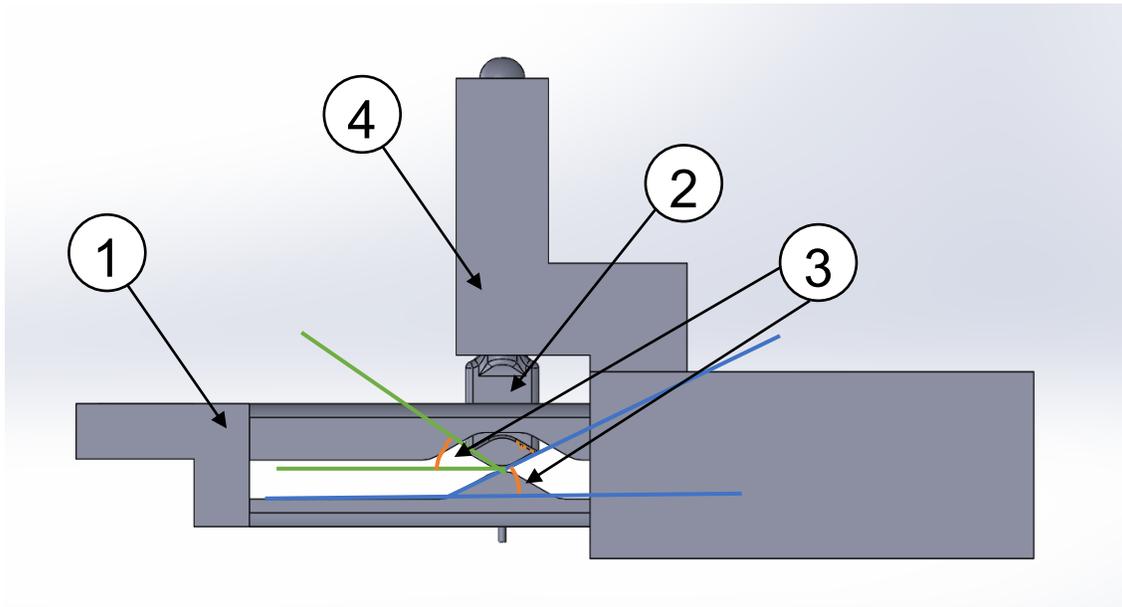


- ① Holes.
- ② Pins.
- ③ Chosen result of experience.

Figure 21. to find right relationship between the hole and the pin.

Next important choice was to choose the right configuration between sliding bar and rising pin. Figure 22 depicts the sliding bar 1 which has to move smoothly within main block 4, in order to save the energy and have chance to choose between smaller and weaker motors with no more voltage supply but 5 Volts. Several experiments have been performed with what the figure should be chosen for the pin's supporter to slide smoothly and the angle of 30 degrees has proven to

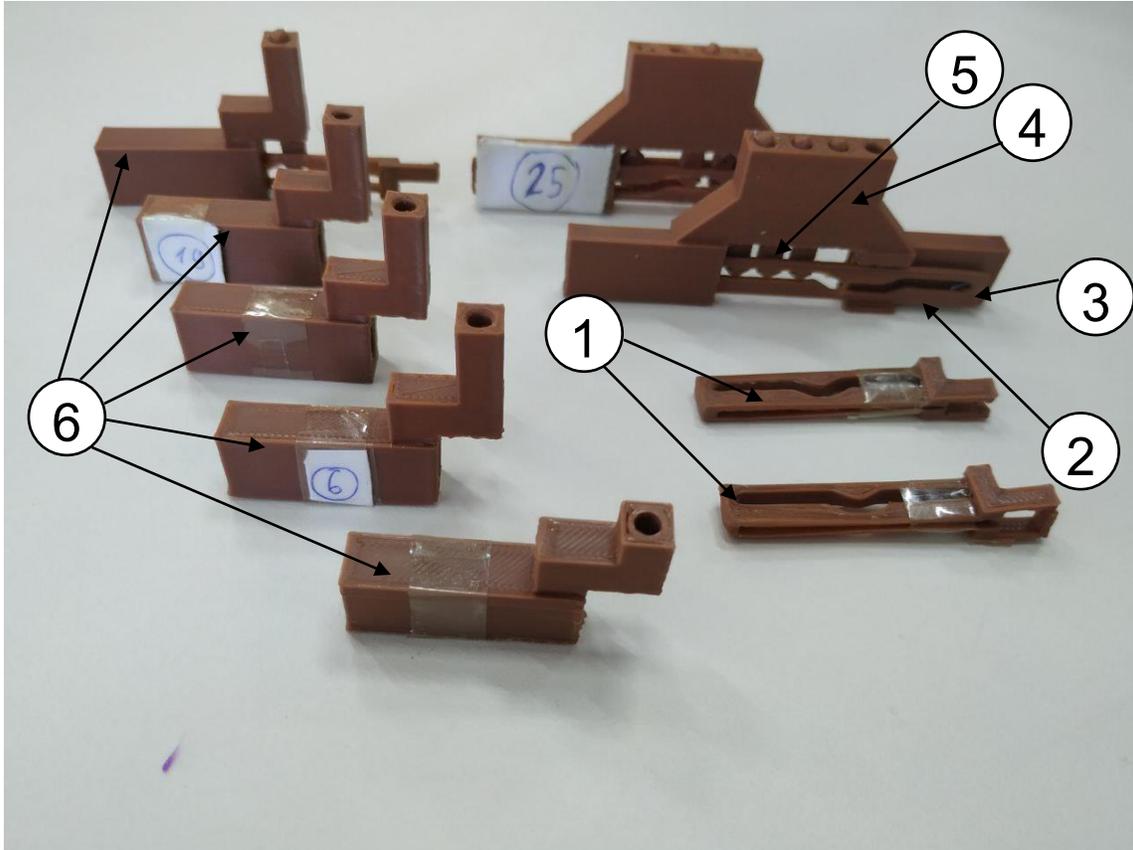
be the one that does not make one pin go out of its geometry (Figure 25). The case is, the ideal angle is those that tends to 0, but practically, the boundary is built by the pin geometry and 30 degrees is the best angle to keep the pin within the boundaries.



- ① Sliding bar.
- ② Rising pin.
- ③ 30 degrees.
- ④ Main block

Figure 22. to choose the right configuration between sliding bar and rising pin

It is not obvious, will this work for rectangular parts (Sliding bars) also and it became right, that for rectangular parts, that relationship changes and horizontal and vertical parts of rectangle differ. For vertical best match could be 0.25, but for horizontal distance (Figure 23) the relation should be not less than 0.5, moving parts not to get stuck.

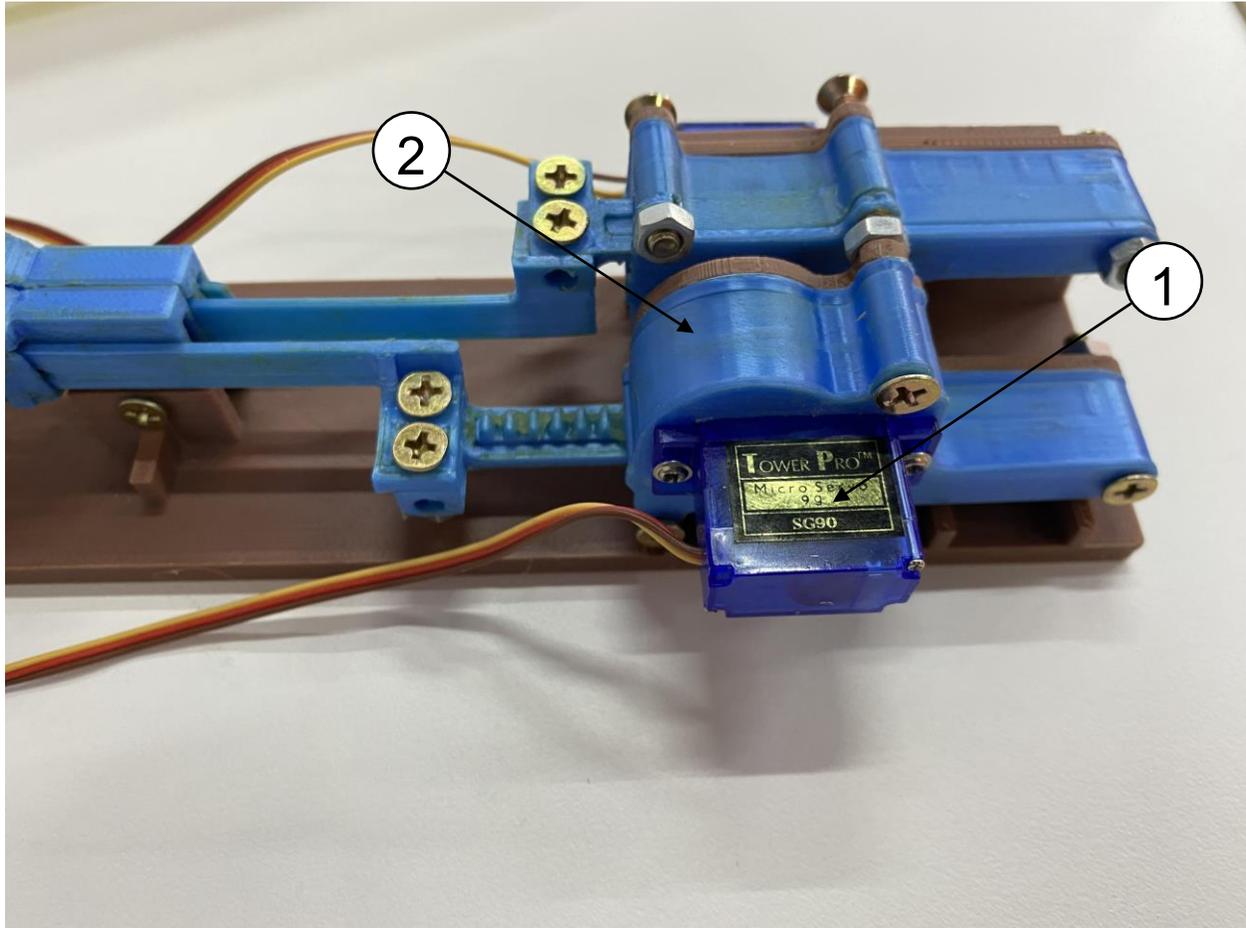


- ① Rectangular parts (Sliding bars).
- ② Horizontal part.
- ③ Vertical part.
- ④ Main block.
- ⑤ Rising pin.
- ⑥ Experiments to choose the right configuration between sliding bar and rising pin.

Figure 23. Main block and Sliding bar.

The mechanism in Figure 23 shows the pin, the main block, and the sliding bar. As the sliding bar moves right and left, the pin sliding (here the name ‘Slider’ comes from) rises and fall back below the zero point of touch. The sliding bar is moved with servo-motor linear actuator (Figure 24).

Only after taking all the numbers and offsets have, we replicated the calculations and ended up with 8 pins Braille Display (Figure 25).



- ① Servo motor.
- ② Linear actuator.

Figure 24. Servo-motor linear actuator.

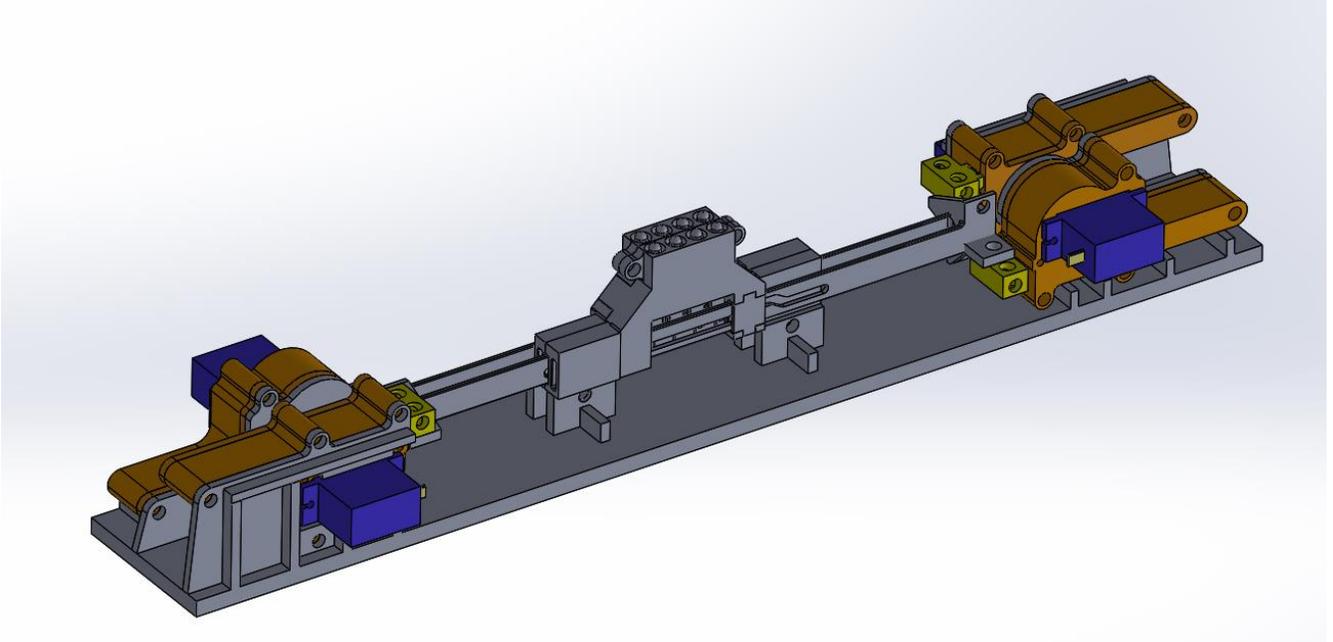
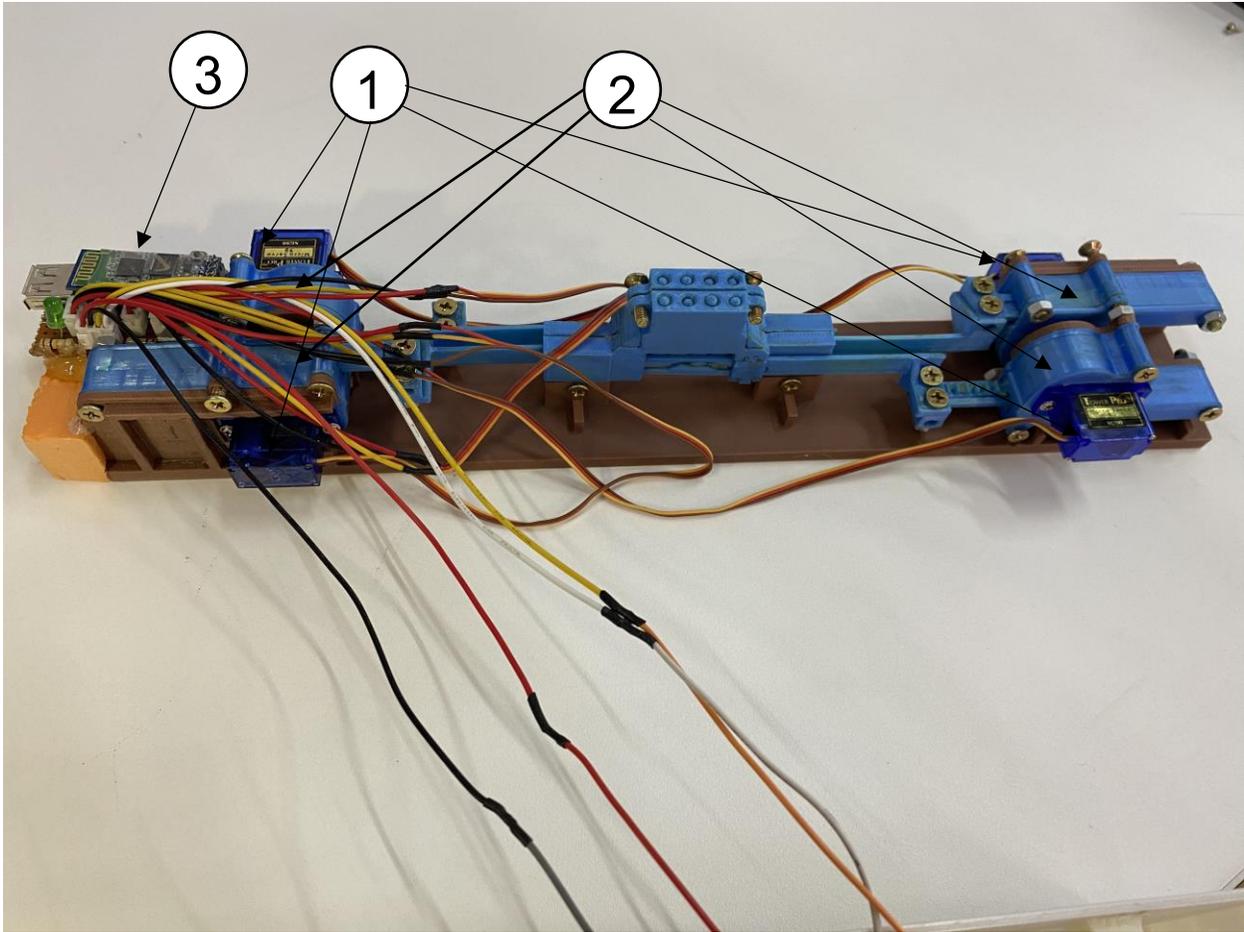


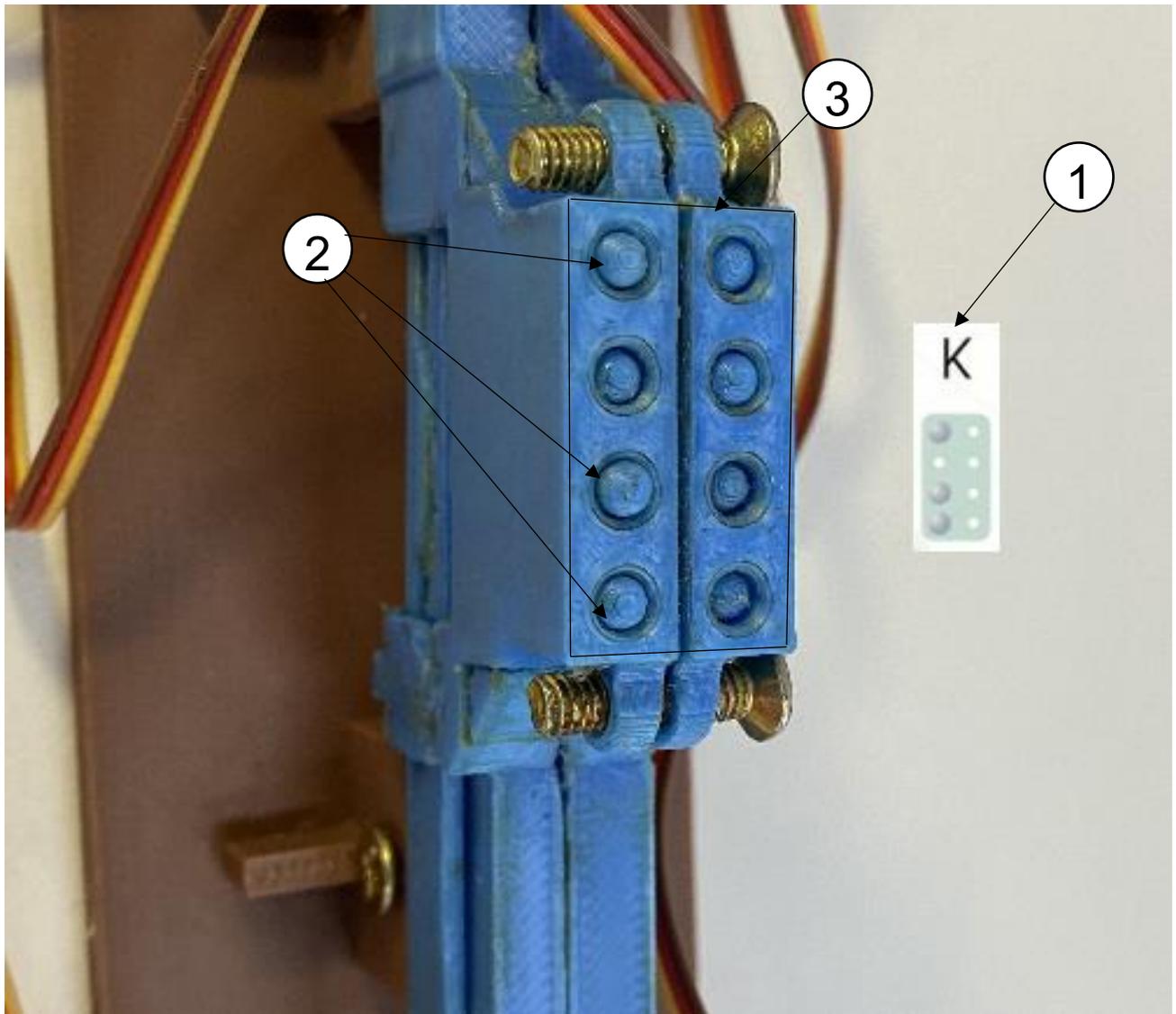
Figure 25. Final construction of 8 pins Braille display in 3d view.

Figure 26 shows the overall view of the braille cell, and it is generally controlled by 4 servomotor based linear actuators and those linear actuators are controlled by ATmega328P - 8-bit AVR Microcontroller.



- ① Servo motors.
- ② Linear actuators.
- ③ ATmega328P - 8-bit AVR Microcontroller. (Located under bluetooth module hc-05).

Figure 26. The overall view of the braille cell.



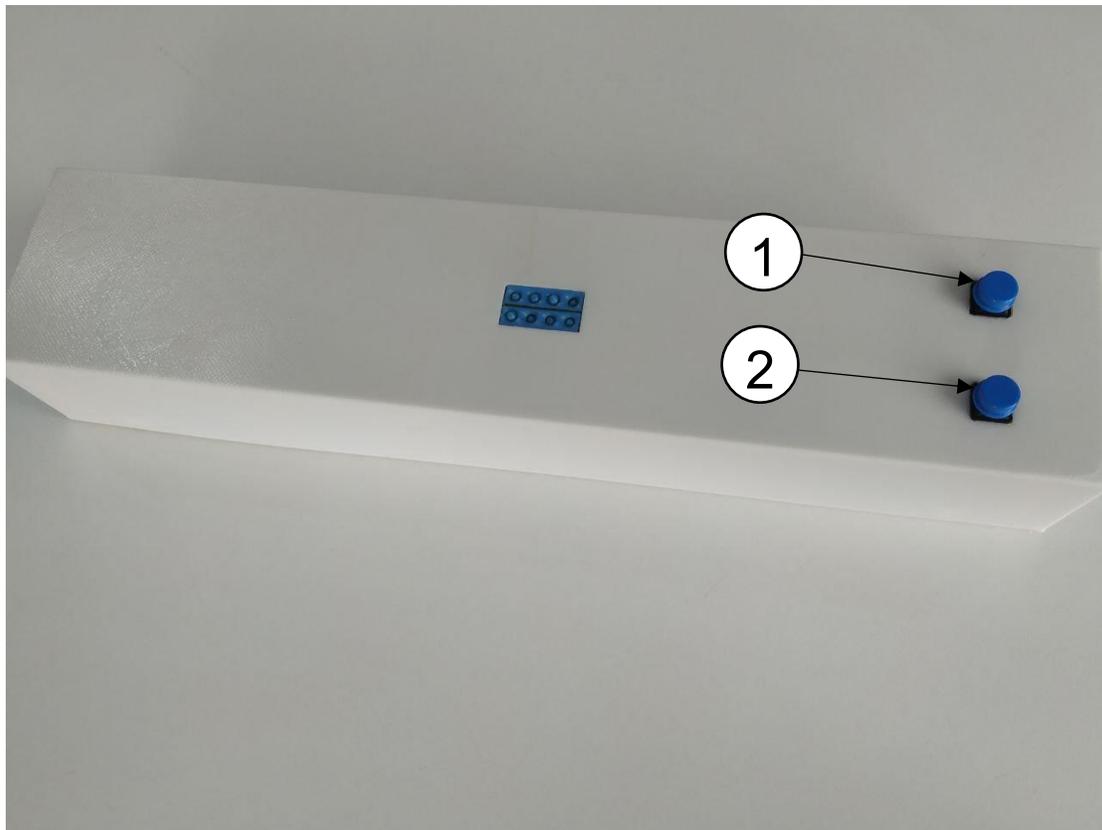
- ① 'K' in Braille Alphabet.
- ② 'K' in Braille cells.
- ③ Braille cells.

Figure 27. Letter 'K' in Braille cells.

Figure 27 shows the letter 'B' implemented with a Braille cell and this cell is the key point of the whole project and that is the part the user will interact with.

The final product has been shown in Figure 28 and the interface consists of the two buttons 'move forward' and 'move back'. The other part of the interface is the

Braille cell that can implement any possible letter within ASCII table (Appendix 4, ASCII).



- ① Move forward button.
- ② Move backward button.

Figure 28. Final view of Braille display.

Currently, that is the last trial of the last Slider i.e., the sixth one and the work is continuing improving the machine.

Holding the price tag in the minimum was the main idea, that is the reason, no design strategy was proposed for the actuator used in the structure. Instead, the thorough learning of the market has been performed to get the most miniature and effective actuator with the minimum price. Further the thesis explains all the steps and actions performed throughout the implementation of the project.

Team of engineers lead the project handing over several times for years in the laboratories and R&D centers. The project started in the Technopark, Center of Mechatronics of Politecnico di Torino Tashkent and with a long gap is continuing in the Robotics and 3D laboratories of the Ministry of Innovative Development with the ones financial support. The research is mostly taken by the alumni of the mentioned school and specifically the alumni of Mechatronics School of the University. The comparative analysis has been performed with data taken empirically and iteratively purchasing the part or printing, then comparing with all the previous results were made, to ensure that no fact has been skipped.

The most popular brands of 3D printers were used, like Ultimaker 2, Flashforge, Ender series, Createbot and their original slicers to ensure the fact that work could be continued in any part of the world with the obtained calculations. For the material used in 3D printer's PLA, as the easiest in printing and holding the most information on the Internet was fed to the machine. Also, for some parts, to decrease the time and material price for the mechanical part, for some parts a laser cutter was used. CAD tools like NX and Solidworks together with AutoCad (for laser cutter) were employed to get the structure of the project. Obviously, personal computers and other peripheral devices were employed.

The electrical part and all of the codes (Appendix 5, Developed codes) has been depicted in the appendices with developed schematics (Appendix 6, Developed schematics) and PCB view of the servo motor controllers. For the conversion of the visual text into the tactile text the usual technology was deployed with code, controller and electrical schematics (Appendix 6, Developed schematics) that has been extensively shown in the appendices.

The goal was to develop a product! So, the weak analysis or unprofessional approaches were immediately rejected by the market itself, leaving no room for maneuvers and bridges to retreat. There are still no software tools and methods to analyze 3D prints, at least, there are no products available in the market to make more or less accurate simulation of 3D printed objects. That is the main reason, empirical and comparative approaches were employed, and they proved to be effective, because a relatively effective prototype has been developed and there is no other team in the world, who could achieve such a low-price tag and effective mechanism, together with electrical and software.

4. Results.

Section description.

Iterative and comparative analysis has given the following results - the best product could be engineered with the following situation in the market of available actuators and capabilities of prototyping technologies together with prices is the sliding mechanism used by the B.R.A.V.E. team of developers with the account in the Thingiverse platform [\[23\]](#).

Material setup

The minimum pin size could be reproduced by the introduced technology, is 3 mm, that is 6 times larger, than the standard one and there is a large field to develop further. The infill used in printing is 50%-70% and 100% for pins only.

The best construction configuration is 30 degrees for the inclination and pin's part that connects to that inclination. One bar can raise a pair of pins, giving 4 possible combinations:

00

01

10

11

And this can give all the characters in accordance with ASCII standards. The reason to use one servo motor is that 9g tower pro servo motor can make 180 degrees of rotation max, otherwise only 3 motors could be used in development.

Pin size standards

It does not fit the internationally accepted braille size standards and are larger for twice as the accepted ones by each side (Figure 29).

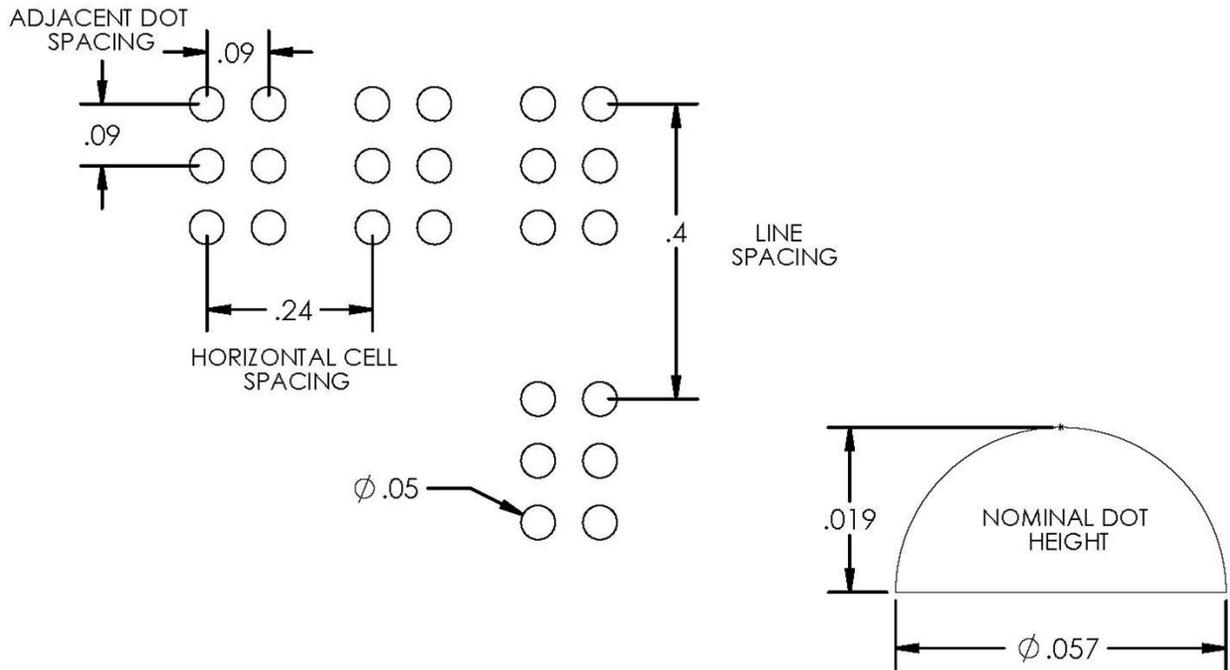


Figure 29. Braille size standards[\[26\]](#).

It was found that it is possible to produce the refreshable Braille displays at an affordable price, using only ordinary FDM 3D printers and other amateur electronics platforms, like Arduino etc.

The outcome is ready to make (DIY) and use design of the affordable refreshable Braille display in most parts of the planet, since it is difficult to find city with no 3D printer and Arduino parts whatsoever.

5. Discussion.

The requirement for more math applied.

There must be done a great deal of work further, with more integration of laws of statistics, analysis, and laws of strength of materials and fundamentals of machine design. There must be further lots of tables with calculations, in order to prove reliability of construction. Research on smooth strength checking and crash testing machines and further analysis is lacking currently and most of the calculation has been obtained with not the most sophisticated methods and measuring devices and most of the concentration has been made on developing the product and reducing the cost that took time.

The sliding mechanism shows the best characteristics that might arise during other approaches - not strong enough, does not hold the finger pressure, cheap and easily producible and easy in assembly.

Discussion of drawbacks of the current solution.

Yet, there are other drawbacks, the project has an intention to work further on - the mechanism is several times slower than the concurrent classical piezo actuator-design displays.

Production process with 3D printers is slow and takes almost one day to produce one item and full engagement of a worker. The process is error prone, but it is not meant to be used in production. But there are almost no cities without 3D printers and makers that would be able to build the item with full instructions given about building.

The construction is not strong enough to come up with inaccurate attitudes and can break through the lines of printing. It is incompatible with strong sunlight and may be deformed easily after a long stay under sunlight. Also, there are lots of parts under continuous friction and could be deformed and wearied out after a period of employment. That is the reason, hard lubricators i.e., greases should be used, and the mechanism should be oiled during some period of time.

The project goal is increasing the reliability of mechanisms presented by the BRAVE team of developers and adding parts that make the dots fall back and contributing cheap servo motors usually met in Arduino kits, that increases the control and eases the development of applications to run the machine, since Arduino is an extremely popular platform. During the project the size of the construction has been changed in order to make it possible print in ordinary FDM 3D printers. That makes it possible for any makers throughout the World to print them in their workshops and sell or use it for their own purposes.

Advantages: The construction developed by the team working on this project is more reliable, since it can resist finger presses (absent in MOBLED design) and does not consume electrical energy in constant mode, but only during changing state from risen to fallen.

The price of the parts and materials are all cheap and easily available in the market and that reduces the price of the most spread-out models by 5 times, making it available to the large number of visually impaired in the developing countries.

There are not so great losses in overall quality and quality loss might be around 20%, but the price falls for several times and this product can find its market.

Returning of mechanism to passive mode is implemented via sliding mechanism, instead of spring in BRAVE team design.

Excess of control is obtained through using servo motors.

6. Conclusion.

The product has been developed consisting of the structure, the electrical part and code. Unfortunately, with the 3D printing technologies available today, it is impossible to hold on with such a tiny Braille Standards (Appendix 2, Braille standards). But, at least, it was possible to keep the scales of the sizes of the parts. For example, the size of one dot in accordance with Braille Standard is 1.5 mm, but the structure in the current work has a dot of 3 mm, i.e., twice the scale (200%) and so has been increased the distance between dots – twice. The most optimum structure with minimum parts has been developed to become a part of the most reduced cost Braille display in the market.

The structure has been developed specifically for FDM 3D printer and all of the sizes and configurations have been arranged to fit the requirements of FDM printing technology, taking into account models of the printers and materials used. The parts have been chosen so that they are available in almost any city with internet or might be purchased easily through the mailing system. The coding platform has been chosen as the most spread out through the world of makers and starting design engineers - Arduino. The electrical part is obvious and could be fixed or assembled by anyone with little or not at all practice in the field.

There are several weak sides of the project and work has to be done to complete with developing the product and to become a part of the best scientific works. More simulation instruments have to be applied to the project as it becomes available in the market of software or as part of CAD we are using currently. There are plans to make some stress-analysis on smooth mechanical machines that unfortunately has been unavailable in the labs we worked and in any lab of the country of residency.

Currently a team of engineers in the National Office for technology transfers are working on developing the interface of the program, that will be easy to use by both the visually impaired and their assistants. The platform is planned to fit into Android, iOS, Windows and MacOS with cross-platform development environment like Flutter, React Native or Xamarin. It is going to be an application with capability of uploading the file or part of the text and continuously convert it into tactile text on the interface of the Braille display. The app is going to work through Bluetooth and WiFi protocols to ensure maximum accuracy in wireless contact.

More calculations will be available in the next step of the project where the SLA printer will be applied with the parts with more solid structure and during development of the press-form for the final product.

7.References.

[0] weblink:

(https://www.google.com/search?q=number+of+blinds+in+the+world&sxsrf=ALeKk03t5RSKqRcPdMXuvtDsyf0g65kIZg%3A1623671510689&ei=1kLHYOSmKdCvrgSh_rGYDw&oq=number+of+blinds+in&gs_lcp=Cgdnd3Mtd2l6EAMYATICCAyBwgAEIcCEBQyBggAEBYQHjoHCAAQRxCwAzoHCCMQ6gIQJzoECCMQJzoICC4QxwEQowI6AgguOggILhDHARCvAToFCAAQkQI6BAgAEEM6CAgAEMkDEJECUMbpBFiDywVgot8FaAFwA3gAgAGDAogB9xiSAQYwLjE1LjSYAQGgAQGqAQdnd3Mtd2l6sAEKyAEIwAEB&sclient=gws-wiz).

[1] weblink: (<https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>).

[1] weblink: (<https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>).

[2] weblink: (<https://www.stronggo.com/blog/etiquette-interacting-blind-person>).

[3] weblink: (<https://www.lighthouseguild.org/vision-health/tactile-illustrated-books-did-you-say-a-little-miracle/>).

[4] weblink: (<https://www.marketplace.org/2017/10/12/braille-versions-textbooks-help-blind-college-students-succeed/>).

[5] weblink: (<https://www.kent-teach.com/Blog/post/2018/01/04/a-brief-history-of-braille-world-braille-day.aspx#:~:text=Braille%20was%20adopted%20by%20France,people%20to%20communicate%20until%201918>).

[6] weblink: (https://en.wikipedia.org/wiki/Slate_and_stylus).

[7] weblink: (<https://www.youtube.com/watch?v=A2bDR6daHW8>).

[8a] weblink: (<https://brailier.perkins.org/collections/perkins-brailers>).

[8b] weblink: (<https://www.boundlessat.com/Braille-Displays>)

[9] weblink: (https://www.researchgate.net/figure/Simplified-schematic-of-a-refreshable-braille-display_fig1_275051459).

[10] weblink: (<https://www.tetragon.at/img/zoom/about-braille/braille-display-closeup.jpg>).

[11] weblink: (<https://thelowvisionstore.com/collections/braille-displays-and-notetakers>).

[12] weblink: (<https://www.dancingdots.com/prodesc/HIMS.htm>).

[13] weblink: (<http://henryomd.blogspot.com/2011/06/braille-cell-control-of-ufluidic-card.html>).

[14] weblink: (<https://store.humanware.com/usa/blindness/braille-displays?limit=all>).

[15] Madaeon, Pradeep Kumar N, “MOLBED 2 Modular Low-cost Braille Electro Display”, hackaday.io, weblink:(<https://hackaday.io/project/27126-molbed-2-modular-low-cost-braille-electro-display>).

[16] A M Muntasir Rahman, Shaker Mahmud Khandaker, Nasif Noor Saleheen, Tasnia Nobi Afee, Naba Afrin, Md. Ashraf Alam, “A Portable Braille Refreshable Display Using Micro Servos”, 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR).

- [17] Syed Adnan Akhtar; Dinkar Prasad, “Braille refreshable display using lead screw actuation”, 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON).
- [18] Fung-HueiYehShih-HaoLiang, “Mechanism design of the flapper actuator in Chinese Braille display”, Sensors and Actuators A: Physical, 2007.
- [19] Joonyeong Kim, Byung-Kil Han, Dongbum Pyo, Semin Ryu, Hanbyeol Kim, Dong-Soo Kwon, “Braille Display for Portable Device Using Flip-Latch Structured Electromagnetic Actuator”, IEEE Transactions on Haptics (Volume: 13, Issue: 1, Jan.-March 1, 2020).
- [20] John Roberts, Oliver Slattery, David Kardos and Brett Swope, “New Technology Enables Many-Fold Reduction in the Cost of Refreshable Braille Displays”, ACM (Association for Computing Machinery) Digital Library, 2000.
- [21] Shahruk Hossain, Abdullah Abyad Raied, Asifur Rahman, Zaowad Rahabin Abdullah, Dipanjan Adhikary, Ahsan Rabby Khan, Arnab Bhattacharjee, Celia Shahnaz, Shaikh Anowarul Fattah, “Text to Braille Scanner with Ultra Low-Cost Refreshable Braille Display”, 2018 IEEE Global Humanitarian Technology Conference (GHTC).
- [22] Raj D. Sutariya, Himanshu S. Singh, Sudhir R. Babariya, Sajid Ali Kadiyar, Darshan H. Modi, “Refreshable Braille Display for the Visually Impaired”, 2017 14th IEEE India Council International Conference (INDICON).
- [23] pyrophreak, “3D Printed Braille Display”, Thingiverse.com, 2013, weblink: (<https://www.thingiverse.com/thing:90144>).
- [24] weblink: (<https://www.aniwaa.com/buyers-guide/3d-printers/the-best-resin-3d-printer-sla-and-dlp/>).
- [25] weblink: (<https://www.amazon.in/Generic-DC5-6V-Inhaled-Electromagnet-Solenoid/dp/B072DV2VPR>).
- [26] weblink: (<https://www.perkins.org/sites/default/files/brailier-schematic1.png>).

8. Appendix.

Appendix 1.

current version of letters combinations has been employed in program development and conversion.

Braille Alphabet																			
with capital letters and numbers																			
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o					
p	q	r	s	t	u	v	w	x	y	ü	z	å	ä	ö					
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O					
P	Q	R	S	T	U	V	W	X	Y	Ü	Z	Å	Ä	Ö					
!	()	*	,	-	.	:	;	?	0	1	2	3	4	5	6	7	8	9

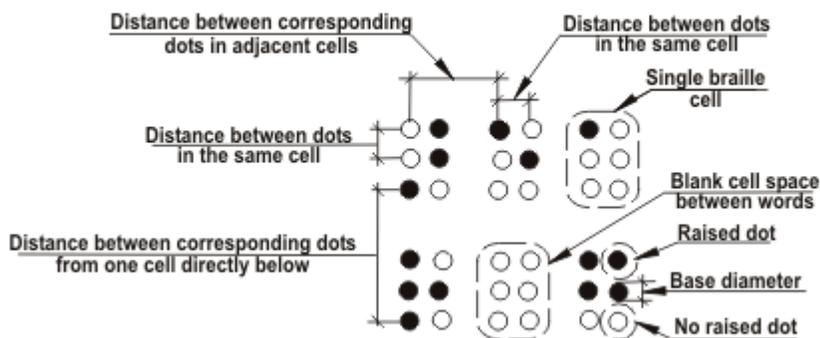
Appendix 2.

Braille standards are the series of embossed dots are evenly arranged in quadrangular letter spaces called cells. A full cell is three dots high and two dots wide, and each cell may contain up to six dots. Standard braille is made up of 6 dots and Unicode renders 8 dot braille.

Table 703.3.1 Dimensions

Measurement range	Minimum in Inches Maximum in Inches
Dot base diameter	0.059 (1.5 mm) to 0.063 (1.6 mm)
Distance between two dots in the same cell ¹	0.090 (2.3 mm) to 0.100 (2.5 mm)
Distance between corresponding dots in adjacent cells ¹	0.241 (6.1 mm) to 0.300 (7.6 mm)
Dot Height	0.025 (0.6 mm) to 0.037 (0.9 mm)
Distance between corresponding dots from one cell directly below ¹	0.395 (10 mm) to 0.400 (10.2 mm)

1. Measured center to center



**Figure 703.3.1
Braille Measurement**

Appendix 3 is very first code developed for Slider II with 8 dots to fit into the standards of ASCII code Braille.

```

1 #include <Servo.h>
2 //Variables
3 Servo ser1;
4 Servo ser2;
5 Servo ser3;
6 Servo ser4;
7 Servo ser5;
8 Servo ser6;
9 Servo ser7;
10 Servo ser8;
11 int nextBut = 22;
12 int prevBut = 23;
13 String sampleText = "Hello"; /*Input texts*/
14 void setup() {
15     // initializing variables
16     ser1.attach(13);
17     ser2.attach(12);
18     ser3.attach(11);
19     ser4.attach(10);
20     ser5.attach(9);
21     ser6.attach(8);
22     ser7.attach(7);
23     ser8.attach(6);
24     pinMode(nextBut, INPUT);
25     pinMode(prevBut, INPUT);
26     /*Access to 'reset' function and
27     resetting all variables*/
28     reset();
29 }
30 void loop() {
31     //Initializing loop variables
32     int i = 0;
33     boolean done = true;
34     //executing inputted text characters
35     do {
36     /*Return to the initial position when
37     the first character was displayed in the text
38     only the previous button is pressed */
39     if (!done && i == 0) {
40         reset();
41         done = true;
42     }
43     /*Converting characters to braille system
44     and displaying it */
45     if (!done) {
46         convert(sampleText[i - 1]);
47         displayChar();
48         done = true;
49     }
50     /*Access to 'next' function
51     and alogirtm for loop variables*/
52     if (next() == HIGH) {
53         done = false;
54         i++;
55     }
56     /*Access to 'prev' function
57     and alogirtm for loop variables*/
58     else if (prev() == HIGH) {
59         done = false;
60         i--;
61     }
62     } while (i <= sampleText.length());
63     /*Access to reset function and
64     resetting all variables*/
65     reset();
66 }
67 // Debouncing buttons
68 boolean debounce(boolean last, int pin) {
69     int current = digitalRead(pin);
70     if (last != current) {
71         delay(5);
72         current = digitalRead(pin);
73     }
74     return current;
75 }
76 //Initialized variables out of loop
77 boolean nextLast = LOW;
78 boolean prevLast = LOW;
79 int state[8];
80 //Controlling next button
81 boolean next() {
82     boolean current = debounce(nextLast, nextBut);
83     if (nextLast == LOW && current == HIGH) {
84         nextLast = current;
85     }
86     return HIGH;
87 }
88 nextLast = current;
89 return LOW;
90 }
91 //Controlling previous button
92 boolean prev() {
93     boolean current = debounce(prevLast, prevBut);
94     if (prevLast == LOW && current == HIGH) {
95         prevLast = current;
96         return HIGH;
97     }
98     prevLast = current;
99     return LOW;
100 }
101 //reset function
102 void reset() {
103     ser2.write(115);
104     ser5.write(118);
105     ser6.write(115);
106     ser1.write(110);
107     ser3.write(112);
108     ser4.write(112);
109     ser7.write(117);
110     ser8.write(120);
111     state[0] = 0;
112     state[1] = 0;
113     state[2] = 0;
114     state[3] = 0;
115     state[4] = 0;
116     state[5] = 0;
117     state[6] = 0;
118     state[7] = 0;
119     delay(100);
120 }
121 /*Converting characters to
122 the braille system by this function*/
123 void convert(char c) {
124     switch (c) {
125         //Small letters
126         case 'a': {
127             state[0] = 1;
128             state[1] = 0;
129             state[2] = 0;
130             state[3] = 0;
131             state[4] = 0;
132             state[5] = 0;
133             state[6] = 0;
134             state[7] = 0;
135             break;
136         }
137         case 'b': {
138             state[0] = 1;
139             state[1] = 1;
140             state[2] = 0;
141             state[3] = 0;
142             state[4] = 0;
143             state[5] = 0;
144             state[6] = 0;
145             state[7] = 0;
146             break;
147         }
148         case 'c': {
149             state[0] = 1;
150             state[1] = 0;
151             state[2] = 0;
152             state[3] = 1;
153             state[4] = 0;
154             state[5] = 0;
155             state[6] = 0;
156             state[7] = 0;
157             break;
158         }
159     }
160     case 'd': {

```

```

161     state[0] = 1;
162     state[1] = 0;
163     state[2] = 0;
164     state[3] = 1;
165     state[4] = 1;
166     state[5] = 0;
167     state[6] = 0;
168     state[7] = 0;
169     break;
170 }
171 case 'e': {
172     state[0] = 1;
173     state[1] = 0;
174     state[2] = 0;
175     state[3] = 0;
176     state[4] = 1;
177     state[5] = 0;
178     state[6] = 0;
179     state[7] = 0;
180     break;
181 }
182 case 'f': {
183     state[0] = 1;
184     state[1] = 1;
185     state[2] = 0;
186     state[3] = 1;
187     state[4] = 0;
188     state[5] = 0;
189     state[6] = 0;
190     state[7] = 0;
191     break;
192 }
193 case 'g': {
194     state[0] = 1;
195     state[1] = 1;
196     state[2] = 0;
197     state[3] = 1;
198     state[4] = 1;
199     state[5] = 0;
200     state[6] = 0;
201     state[7] = 0;
202     break;
203 }
204 case 'h': {
205     state[0] = 1;
206     state[1] = 1;
207     state[2] = 0;
208     state[3] = 0;
209     state[4] = 1;
210     state[5] = 0;
211     state[6] = 0;
212     state[7] = 0;
213     break;
214 }
215 case 'i': {
216     state[0] = 0;
217     state[1] = 1;
218     state[2] = 0;
219     state[3] = 1;
220     state[4] = 0;
221     state[5] = 0;
222     state[6] = 0;
223     state[7] = 0;
224     break;
225 }
226 case 'j': {
227     state[0] = 0;
228     state[1] = 1;
229     state[2] = 0;
230     state[3] = 1;
231     state[4] = 1;
232     state[5] = 0;
233     state[6] = 0;
234     state[7] = 0;
235     break;
236 }
237 case 'k': {
238     state[0] = 1;
239     state[1] = 0;
240     state[2] = 1;
241     state[3] = 0;
242     state[4] = 0;
243     state[5] = 0;
244     state[6] = 0;
245     state[7] = 0;
246     break;
247 }
248 case 'l': {
249     state[0] = 1;
250     state[1] = 1;
251     state[2] = 1;
252     state[3] = 0;
253     state[4] = 0;
254     state[5] = 0;
255     state[6] = 0;
256     state[7] = 0;
257     break;
258 }
259 case 'm': {
260     state[0] = 1;
261     state[1] = 0;
262     state[2] = 1;
263     state[3] = 1;
264     state[4] = 0;
265     state[5] = 0;
266     state[6] = 0;
267     state[7] = 0;
268     break;
269 }
270 case 'n': {
271     state[0] = 1;
272     state[1] = 0;
273     state[2] = 1;
274     state[3] = 1;
275     state[4] = 1;
276     state[5] = 0;
277     state[6] = 0;
278     state[7] = 0;
279     break;
280 }
281 case 'o': {
282     state[0] = 1;
283     state[1] = 0;
284     state[2] = 1;
285     state[3] = 0;
286     state[4] = 1;
287     state[5] = 0;
288     state[6] = 0;
289     state[7] = 0;
290     break;
291 }
292 case 'p': {
293     state[0] = 1;
294     state[1] = 1;
295     state[2] = 1;
296     state[3] = 1;
297     state[4] = 0;
298     state[5] = 0;
299     state[6] = 0;
300     state[7] = 0;
301     break;
302 }
303 case 'q': {
304     state[0] = 1;
305     state[1] = 1;
306     state[2] = 1;
307     state[3] = 1;
308     state[4] = 1;
309     state[5] = 0;
310     state[6] = 0;
311     state[7] = 0;
312     break;
313 }
314 case 'r': {
315     state[0] = 1;
316     state[1] = 1;
317     state[2] = 1;
318     state[3] = 0;
319     state[4] = 1;
320     state[5] = 0;
321     state[6] = 0;
322     state[7] = 0;
323     break;
324 }
325 case 's': {
326     state[0] = 0;
327     state[1] = 1;
328     state[2] = 1;
329     state[3] = 1;
330     state[4] = 0;
331     state[5] = 0;
332     state[6] = 0;
333     state[7] = 0;
334     break;
335 }
336 case 't': {
337     state[0] = 0;
338     state[1] = 1;
339     state[2] = 1;
340     state[3] = 1;
341     state[4] = 1;
342     state[5] = 0;
343     state[6] = 0;
344     state[7] = 0;
345     break;
346 }
347 case 'u': {
348     state[0] = 1;
349     state[1] = 0;
350     state[2] = 1;
351     state[3] = 0;
352     state[4] = 0;
353     state[5] = 1;
354     state[6] = 0;
355     state[7] = 0;
356     break;
357 }
358 case 'v': {
359     state[0] = 1;
360     state[1] = 1;
361     state[2] = 1;
362     state[3] = 0;
363     state[4] = 0;
364     state[5] = 1;
365     state[6] = 0;
366     state[7] = 0;
367     break;
368 }
369 case 'w': {
370     state[0] = 0;
371     state[1] = 1;
372     state[2] = 0;
373     state[3] = 1;
374     state[4] = 1;
375     state[5] = 1;
376     state[6] = 0;
377     state[7] = 0;
378     break;
379 }
380 case 'x': {
381     state[0] = 1;
382     state[1] = 0;
383     state[2] = 1;
384     state[3] = 1;
385     state[4] = 0;
386     state[5] = 1;
387     state[6] = 0;
388     state[7] = 0;
389     break;
390 }
391 case 'y': {
392     state[0] = 1;
393     state[1] = 0;
394     state[2] = 1;
395     state[3] = 1;
396     state[4] = 1;
397     state[5] = 1;
398     state[6] = 0;
399     state[7] = 0;
400     break;

```

```

481     }
482     case 'D': {
483         state[0] = 1;
484         state[1] = 0;
485         state[2] = 0;
486         state[3] = 1;
487         state[4] = 1;
488         state[5] = 0;
489         state[6] = 1;
490         state[7] = 0;
491         break;
492     }
493     case 'E': {
494         state[0] = 1;
495         state[1] = 0;
496         state[2] = 0;
497         state[3] = 0;
498         state[4] = 1;
499         state[5] = 0;
500         state[6] = 1;
501         state[7] = 0;
502         break;
503     }
504     case 'F': {
505         state[0] = 1;
506         state[1] = 1;
507         state[2] = 0;
508         state[3] = 1;
509         state[4] = 0;
510         state[5] = 0;
511         state[6] = 1;
512         state[7] = 0;
513         break;
514     }
515     case 'G': {
516         state[0] = 1;
517         state[1] = 1;
518         state[2] = 0;
519         state[3] = 1;
520         state[4] = 1;
521         state[5] = 0;
522         state[6] = 1;
523         state[7] = 0;
524         break;
525     }
526     case 'H': {
527         state[0] = 1;
528         state[1] = 1;
529         state[2] = 0;
530         state[3] = 0;
531         state[4] = 1;
532         state[5] = 0;
533         state[6] = 1;
534         state[7] = 0;
535         break;
536     }
537     case 'I': {
538         state[0] = 0;
539         state[1] = 1;
540         state[2] = 0;
541         state[3] = 1;
542         state[4] = 0;
543         state[5] = 0;
544         state[6] = 1;
545         state[7] = 0;
546         break;
547     }
548     case 'J': {
549         state[0] = 0;
550         state[1] = 1;
551         state[2] = 0;
552         state[3] = 1;
553         state[4] = 1;
554         state[5] = 0;
555         state[6] = 1;
556         state[7] = 0;
557         break;
558     }
559     case 'K': {
560         state[0] = 1;
561         state[1] = 0;
562         state[2] = 1;
563         state[3] = 0;
564         state[4] = 0;
565         state[5] = 0;
566         state[6] = 1;
567         state[7] = 0;
568         break;
569     }
570     case 'L': {
571         state[0] = 1;
572         state[1] = 1;
573         state[2] = 1;
574         state[3] = 0;
575         state[4] = 0;
576         state[5] = 0;
577         state[6] = 1;
578         state[7] = 0;
579         break;
580     }
581     case 'M': {
582         state[0] = 1;
583         state[1] = 0;
584         state[2] = 1;
585         state[3] = 1;
586         state[4] = 0;
587         state[5] = 0;
588         state[6] = 1;
589         state[7] = 0;
590         break;
591     }
592     case 'N': {
593         state[0] = 1;
594         state[1] = 0;
595         state[2] = 1;
596         state[3] = 1;
597         state[4] = 1;
598         state[5] = 0;
599         state[6] = 1;
600         state[7] = 0;
601         break;
602     }
603     case 'O': {
604         state[0] = 1;
605         state[1] = 0;
606         state[2] = 1;
607         state[3] = 0;
608         state[4] = 1;
609         state[5] = 0;
610         state[6] = 1;
611         state[7] = 0;
612         break;
613     }
614     case 'P': {
615         state[0] = 1;
616         state[1] = 1;
617         state[2] = 1;
618         state[3] = 1;
619         state[4] = 0;
620         state[5] = 0;
621         state[6] = 1;
622         state[7] = 0;
623         break;
624     }
625     case 'Q': {
626         state[0] = 1;
627         state[1] = 1;
628         state[2] = 1;
629         state[3] = 1;
630         state[4] = 1;
631         state[5] = 0;
632         state[6] = 1;
633         state[7] = 0;
634         break;
635     }
636     case 'R': {
637         state[0] = 1;
638         state[1] = 1;
639         state[2] = 1;
640         state[3] = 0;
641         state[4] = 1;
642         state[5] = 0;
643         state[6] = 1;
644         state[7] = 0;
645         break;
646     }
647     case 'S': {
648         state[0] = 0;
649         state[1] = 1;
650         state[2] = 1;
651         state[3] = 1;
652         state[4] = 0;
653         state[5] = 0;
654         state[6] = 1;
655         state[7] = 0;
656         break;
657     }
658     case 'T': {
659         state[0] = 0;
660         state[1] = 1;
661         state[2] = 1;
662         state[3] = 1;
663         state[4] = 1;
664         state[5] = 0;
665         state[6] = 1;
666         state[7] = 0;
667         break;
668     }
669     case 'U': {
670         state[0] = 1;
671         state[1] = 0;
672         state[2] = 1;
673         state[3] = 0;
674         state[4] = 0;
675         state[5] = 1;
676         state[6] = 1;
677         state[7] = 0;
678         break;
679     }
680     case 'V': {
681         state[0] = 1;
682         state[1] = 1;
683         state[2] = 1;
684         state[3] = 0;
685         state[4] = 0;
686         state[5] = 1;
687         state[6] = 1;
688         state[7] = 0;
689         break;
690     }
691     case 'W': {
692         state[0] = 0;
693         state[1] = 1;
694         state[2] = 0;
695         state[3] = 1;
696         state[4] = 1;
697         state[5] = 1;
698         state[6] = 1;
699         state[7] = 0;
700         break;
701     }
702     case 'X': {
703         state[0] = 1;
704         state[1] = 0;
705         state[2] = 1;
706         state[3] = 1;
707         state[4] = 0;
708         state[5] = 1;
709         state[6] = 1;
710         state[7] = 0;
711         break;
712     }
713     case 'Y': {
714         state[0] = 1;
715         state[1] = 0;
716         state[2] = 1;
717         state[3] = 1;
718         state[4] = 1;
719         state[5] = 1;
720         state[6] = 1;

```

```

721     state[7] = 0;
722     break;
723 }
724 case '2': {
725     state[0] = 1;
726     state[1] = 0;
727     state[2] = 1;
728     state[3] = 0;
729     state[4] = 1;
730     state[5] = 1;
731     state[6] = 1;
732     state[7] = 0;
733     break;
734 }
735 case 'Ã': {
736     state[0] = 0;
737     state[1] = 0;
738     state[2] = 1;
739     state[3] = 1;
740     state[4] = 1;
741     state[5] = 0;
742     state[6] = 1;
743     state[7] = 1;
744     break;
745 }
746 case 'È': {
747     state[0] = 1;
748     state[1] = 1;
749     state[2] = 0;
750     state[3] = 1;
751     state[4] = 0;
752     state[5] = 1;
753     state[6] = 1;
754     state[7] = 1;
755     break;
756 }
757 case 'É': {
758     state[0] = 1;
759     state[1] = 1;
760     state[2] = 1;
761     state[3] = 1;
762     state[4] = 1;
763     state[5] = 1;
764     state[6] = 1;
765     state[7] = 1;
766     break;
767 }
768 // Numbers
769 case '0': {
770     state[0] = 0;
771     state[1] = 0;
772     state[2] = 1;
773     state[3] = 1;
774     state[4] = 1;
775     state[5] = 1;
776     state[6] = 0;
777     state[7] = 0;
778     break;
779 }
780 case '1': {
781     state[0] = 1;
782     state[1] = 0;
783     state[2] = 0;
784     state[3] = 0;
785     state[4] = 0;
786     state[5] = 1;
787     state[6] = 0;
788     state[7] = 0;
789     break;
790 }
791 case '2': {
792     state[0] = 1;
793     state[1] = 1;
794     state[2] = 0;
795     state[3] = 0;
796     state[4] = 0;
797     state[5] = 1;
798     state[6] = 0;
799     state[7] = 0;
800     break;
801 }
802 case '3': {
803     state[0] = 1;
804     state[1] = 0;
805     state[2] = 0;
806     state[3] = 1;
807     state[4] = 0;
808     state[5] = 1;
809     state[6] = 0;
810     state[7] = 0;
811     break;
812 }
813 case '4': {
814     state[0] = 1;
815     state[1] = 0;
816     state[2] = 0;
817     state[3] = 1;
818     state[4] = 1;
819     state[5] = 1;
820     state[6] = 0;
821     state[7] = 0;
822     break;
823 }
824 case '5': {
825     state[0] = 1;
826     state[1] = 0;
827     state[2] = 0;
828     state[3] = 0;
829     state[4] = 1;
830     state[5] = 1;
831     state[6] = 0;
832     state[7] = 0;
833     break;
834 }
835 case '6': {
836     state[0] = 1;
837     state[1] = 1;
838     state[2] = 0;
839     state[3] = 1;
840     state[4] = 0;
841     state[5] = 1;
842     state[6] = 0;
843     state[7] = 0;
844     break;
845 }
846 case '7': {
847     state[0] = 1;
848     state[1] = 1;
849     state[2] = 0;
850     state[3] = 1;
851     state[4] = 1;
852     state[5] = 1;
853     state[6] = 0;
854     state[7] = 0;
855     break;
856 }
857 case '8': {
858     state[0] = 1;
859     state[1] = 1;
860     state[2] = 0;
861     state[3] = 0;
862     state[4] = 1;
863     state[5] = 1;
864     state[6] = 0;
865     state[7] = 0;
866     break;
867 }
868 case '9': {
869     state[0] = 0;
870     state[1] = 1;
871     state[2] = 0;
872     state[3] = 1;
873     state[4] = 0;
874     state[5] = 1;
875     state[6] = 0;
876     state[7] = 0;
877     break;
878 }
879 }
880 // Punctuation
881 case ' ': {
882     state[0] = 0;
883     state[1] = 0;
884     state[2] = 0;
885     state[3] = 0;
886     state[4] = 0;
887     state[5] = 0;
888     state[6] = 0;
889     state[7] = 0;
890     break;
891 }
892 case ',': {
893     state[0] = 0;
894     state[1] = 1;
895     state[2] = 0;
896     state[3] = 0;
897     state[4] = 0;
898     state[5] = 0;
899     state[6] = 0;
900     state[7] = 0;
901     break;
902 }
903 case '.': {
904     state[0] = 0;
905     state[1] = 0;
906     state[2] = 1;
907     state[3] = 0;
908     state[4] = 0;
909     state[5] = 0;
910     state[6] = 0;
911     state[7] = 0;
912     break;
913 }
914 case '!': {
915     state[0] = 0;
916     state[1] = 0;
917     state[2] = 0;
918     state[3] = 0;
919     state[4] = 1;
920     state[5] = 0;
921     state[6] = 0;
922     state[7] = 0;
923     break;
924 }
925 case '?': {
926     state[0] = 0;
927     state[1] = 1;
928     state[2] = 0;
929     state[3] = 0;
930     state[4] = 0;
931     state[5] = 1;
932     state[6] = 0;
933     state[7] = 0;
934     break;
935 }
936 case ';': {
937     state[0] = 0;
938     state[1] = 1;
939     state[2] = 1;
940     state[3] = 0;
941     state[4] = 0;
942     state[5] = 0;
943     state[6] = 0;
944     state[7] = 0;
945     break;
946 }
947 case ':': {
948     state[0] = 0;
949     state[1] = 1;
950     state[2] = 0;
951     state[3] = 0;
952     state[4] = 1;
953     state[5] = 0;
954     state[6] = 0;
955     state[7] = 0;
956     break;
957 }
958 case '"': {
959     state[0] = 0;
960 }

```

```

961     state[1] = 0;
962     state[2] = 0;
963     state[3] = 1;
964     state[4] = 0;
965     state[5] = 0;
966     state[6] = 0;
967     state[7] = 0;
968     break;
969 }
970
971 case '(': {
972     state[0] = 0;
973     state[1] = 1;
974     state[2] = 1;
975     state[3] = 0;
976     state[4] = 0;
977     state[5] = 1;
978     state[6] = 0;
979     state[7] = 0;
980     break;
981 }
982 case ')': {
983     state[0] = 0;
984     state[1] = 0;
985     state[2] = 1;
986     state[3] = 0;
987     state[4] = 1;
988     state[5] = 1;
989     state[6] = 0;
990     state[7] = 0;
991     break;
992 }
993 case '[': {
994     state[0] = 1;
995     state[1] = 1;
996     state[2] = 1;
997     state[3] = 0;
998     state[4] = 1;
999     state[5] = 1;
1000    state[6] = 0;
1001    state[7] = 0;
1002    break;
1003 }
1004 case ']': {
1005     state[0] = 0;
1006     state[1] = 1;
1007     state[2] = 1;
1008     state[3] = 1;
1009     state[4] = 1;
1010     state[5] = 1;
1011     state[6] = 0;
1012     state[7] = 0;
1013     break;
1014 }
1015 case '{': {
1016     state[0] = 1;
1017     state[1] = 1;
1018     state[2] = 1;
1019     state[3] = 0;
1020     state[4] = 1;
1021     state[5] = 1;
1022     state[6] = 1;
1023     state[7] = 0;
1024     break;
1025 }
1026 case '}': {
1027     state[0] = 0;
1028     state[1] = 1;
1029     state[2] = 1;
1030     state[3] = 1;
1031     state[4] = 1;
1032     state[5] = 1;
1033     state[6] = 1;
1034     state[7] = 0;
1035     break;
1036 }
1037 case '/': {
1038     state[0] = 1;
1039     state[1] = 1;
1040     state[2] = 1;
1041     state[3] = 1;
1042     state[4] = 1;
1043     state[5] = 1;
1044     state[6] = 1;
1045     state[7] = 1;
1046     break;
1047 }
1048 }
1049 }
1050 /*Displaying character which is
1051 converted to braille system*/
1052 void displayChar() {
1053     if (state[0]) //1st servo
1054     {
1055         ser1.write(65);
1056     }
1057     else
1058     {
1059         ser1.write(115);
1060     }
1061     if (state[1]) //2nd servo
1062     {
1063         ser2.write(75);
1064     }
1065     else
1066     {
1067         ser2.write(115);
1068     }
1069     if (state[2]) //3rd servo
1070     {
1071         ser3.write(65);
1072     }
1073     else
1074     {
1075         ser3.write(112);
1076     }
1077     if (state[3]) //4th servo
1078     {
1079         ser4.write(66);
1080     }
1081     else
1082     {
1083         ser4.write(112);
1084     }
1085     if (state[4]) //5th servo
1086     {
1087         ser5.write(70);
1088     }
1089     else
1090     {
1091         ser5.write(118);
1092     }
1093     if (state[5]) //6th servo
1094     {
1095         ser6.write(70);
1096     }
1097     else
1098     {
1099         ser6.write(110);
1100     }
1101     if (state[6]) //7th servo
1102     {
1103         ser7.write(65);
1104     }
1105     else
1106     {
1107         ser7.write(117);
1108     }
1109     if (state[7]) //8th servo
1110     {
1111         ser8.write(60);
1112     }
1113     else
1114     {
1115         ser8.write(120);
1116     }
1117     delay(100);
1118 }

```

Appendix 4.

ASCII table the principle of which has been used in this project while replicating textual visual data into tactile.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Appendix 5. here the code of Slider VI has been depicted, all the functionalities has been shown in the comments inside the program.

```

1 #include <Servo.h>
2 //Variables
3 Servo ser1;
4 Servo ser2;
5 Servo ser3;
6 Servo ser4;
7 int nextBut = 2;
8 int prevBut = 4;
9 String sampleText;
10 void setup() {
11     // initializing variables
12     Serial.begin(9600);
13     ser1.attach(6);
14     ser2.attach(9);
15     ser3.attach(10);
16     ser4.attach(11);
17     pinMode(nextBut, INPUT);
18     pinMode(prevBut, INPUT);
19     /*Access to 'reset' function and
20      resetting all variables*/
21     reset();
22 }
23 void loop() {
24     //Initializing loop variables
25     int i = 0;
26     boolean done = true;
27     /*To allows input from console to program
28      via "Serial monitor"*/
29     while (Serial.available() == 0)
30     {
31         reset();
32     }
33     /*read string from serial monitor*/
34     String sampleText = Serial.readString();
35     //executing inputted text characters
36     do {
37         /*After reading string from serial monitor
38          it resets motors to initial position via stopping loop*/
39         if (Serial.available() > 0)
40         {
41             break;
42         }
43         /*Return to the initial position when
44          the first character was displayed in the text
45          only the previous button is pressed */
46         if (!done && i == 0) {
47             reset();
48             done = true;
49         }
50         /*Converting characters to braille system
51          and displaying it */
52         if ('done') {
53             convert(sampleText[i - 1]);
54             displayChar();
55             done = true;
56         }
57         /*Access to 'next' function
58          and alogirtm for loop variables*/
59         if (next() == HIGH) {
60             done = false;
61             i++;
62         }
63         /*Access to 'prev' function
64          and alogirtm for loop variables*/
65         else if (prev() == HIGH) {
66             done = false;
67             i--;
68         }
69     } while (i <= sampleText.length());
70     /*Access to reset function and
71      resetting all variables*/
72     reset();
73 }
74 // Debouncing buttons
75 boolean debounce(boolean last, int pin) {
76     int current = digitalRead(pin);
77     if (last != current) {
78         delay(5);
79         current = digitalRead(pin);
80     }
81     return current;
82 }
83 //Initialized variables out of loop
84 boolean nextLast = LOW;
85 boolean prevLast = LOW;
86 int state[12];
87 //Controlling next button
88 boolean next() {
89     boolean current = debounce(nextLast, nextBut);
90     if (nextLast == LOW && current == HIGH) {
91         nextLast = current;
92     }
93     return HIGH;
94 }
95 nextLast = current;
96 return LOW;
97 }
98 //Controlling previous button
99 boolean prev() {
100    boolean current = debounce(prevLast, prevBut);
101    if (prevLast == LOW && current == HIGH) {
102        prevLast = current;
103        return HIGH;
104    }
105    prevLast = current;
106    return LOW;
107 }
108 //reset function
109 void reset() {
110     ser1.write(20);
111     ser2.write(170);
112     ser3.write(130);
113     ser4.write(0);
114
115     state[0] = 0;
116     state[1] = 0;
117     state[2] = 0;
118     state[3] = 0;
119     state[4] = 0;
120     state[5] = 0;
121     state[6] = 0;
122     state[7] = 0;
123     state[8] = 0;
124     state[9] = 0;
125     state[10] = 0;
126     state[11] = 0;
127     delay(100);
128 }
129 /*Converting characters to
130 the braille system by this function*/
131 void convert(char c) {
132     switch (c) {
133         //Small letters
134         case 'a': {
135             state[0] = 1; //1st servo
136             state[1] = 0;
137             state[2] = 0;
138
139             state[3] = 0; //2nd servo
140             state[4] = 0;
141             state[5] = 0;
142
143             state[6] = 0; //3rd servo
144             state[7] = 0;
145             state[8] = 0;
146
147             state[9] = 0; //4th servo
148             state[10] = 0;
149             state[11] = 0;
150             break;
151         }
152         case 'b': {
153             state[0] = 0; //1st servo
154             state[1] = 1;
155             state[2] = 0;
156
157             state[3] = 0; //2nd servo
158             state[4] = 0;
159             state[5] = 0;
160         }

```

```

161     state[6] = 0; //3rd servo
162     state[7] = 0;
163     state[8] = 0;
164
165     state[9] = 0; //4th servo
166     state[10] = 0;
167     state[11] = 0;
168     break;
169 }
170 case 'c': {
171     state[0] = 1; //1st servo
172     state[1] = 0;
173     state[2] = 0;
174
175     state[3] = 0; //2nd servo
176     state[4] = 0;
177     state[5] = 0;
178
179     state[6] = 1; //3rd servo
180     state[7] = 0;
181     state[8] = 0;
182
183     state[9] = 0; //4th servo
184     state[10] = 0;
185     state[11] = 0;
186     break;
187 }
188 case 'd': {
189     state[0] = 1; //1st servo
190     state[1] = 0;
191     state[2] = 0;
192
193     state[3] = 0; //2nd servo
194     state[4] = 0;
195     state[5] = 0;
196
197     state[6] = 0; //3rd servo
198     state[7] = 1;
199     state[8] = 0;
200
201     state[9] = 0; //4th servo
202     state[10] = 0;
203     state[11] = 0;
204     break;
205 }
206 case 'e': {
207     state[0] = 1; //1st servo
208     state[1] = 0;
209     state[2] = 0;
210
211     state[3] = 0; //2nd servo
212     state[4] = 0;
213     state[5] = 0;
214
215     state[6] = 0; //3rd servo
216     state[7] = 0;
217     state[8] = 1 ;
218
219     state[9] = 0; //4th servo
220     state[10] = 0;
221     state[11] = 0;
222     break;
223 }
224 case 'f': {
225
226     state[0] = 0; //1st servo
227     state[1] = 1;
228     state[2] = 0;
229
230     state[3] = 0; //2nd servo
231     state[4] = 0;
232     state[5] = 0;
233
234     state[6] = 1; //3rd servo
235     state[7] = 0;
236     state[8] = 0 ;
237
238     state[9] = 0; //4th servo
239     state[10] = 0;
240     state[11] = 0;
241     break;
242 }
243 case 'g': {
244
245     state[0] = 0; //1st servo
246     state[1] = 1;
247     state[2] = 0;
248
249     state[3] = 0; //2nd servo
250     state[4] = 0;
251     state[5] = 0;
252
253     state[6] = 0; //3rd servo
254     state[7] = 1;
255     state[8] = 0 ;
256
257     state[9] = 0; //4th servo
258     state[10] = 0;
259     state[11] = 0;
260     break;
261 }
262 case 'h': {
263
264     state[0] = 0; //1st servo
265     state[1] = 1;
266     state[2] = 0;
267
268     state[3] = 0; //2nd servo
269     state[4] = 0;
270     state[5] = 0;
271
272     state[6] = 0; //3rd servo
273     state[7] = 0;
274     state[8] = 1 ;
275
276     state[9] = 0; //4th servo
277     state[10] = 0;
278     state[11] = 0;
279     break;
280 }
281 case 'i': {
282
283     state[0] = 0; //1st servo
284     state[1] = 0;
285     state[2] = 1;
286
287     state[3] = 0; //2nd servo
288     state[4] = 0;
289     state[5] = 0;
290
291     state[6] = 1; //3rd servo
292     state[7] = 0;
293     state[8] = 0 ;
294
295     state[9] = 0; //4th servo
296     state[10] = 0;
297     state[11] = 0;
298     break;
299 }
300 case 'j': {
301
302     state[0] = 0; //1st servo
303     state[1] = 0;
304     state[2] = 1;
305
306     state[3] = 0; //2nd servo
307     state[4] = 0;
308     state[5] = 0;
309
310     state[6] = 0; //3rd servo
311     state[7] = 1;
312     state[8] = 0 ;
313
314     state[9] = 0; //4th servo
315     state[10] = 0;
316     state[11] = 0;
317     break;
318 }
319 case 'k': {
320

```

```

321     state[0] = 1; //1st servo
322     state[1] = 0;
323     state[2] = 0;
324
325     state[3] = 0; //2nd servo
326     state[4] = 0;
327     state[5] = 1;
328
329     state[6] = 0; //3rd servo
330     state[7] = 0;
331     state[8] = 0 ;
332
333     state[9] = 0; //4th servo
334     state[10] = 0;
335     state[11] = 0;
336     break;
337 }
338 case 'l': {
339
340     state[0] = 0; //1st servo
341     state[1] = 1;
342     state[2] = 0;
343
344     state[3] = 0; //2nd servo
345     state[4] = 0;
346     state[5] = 1;
347
348     state[6] = 0; //3rd servo
349     state[7] = 0;
350     state[8] = 0 ;
351
352     state[9] = 0; //4th servo
353     state[10] = 0;
354     state[11] = 0;
355     break;
356 }
357 case 'm': {
358
359     state[0] = 1; //1st servo
360     state[1] = 0;
361     state[2] = 0;
362
363     state[3] = 0; //2nd servo
364     state[4] = 0;
365     state[5] = 1;
366
367     state[6] = 1; //3rd servo
368     state[7] = 0;
369     state[8] = 1 ;
370
371     state[9] = 0; //4th servo
372     state[10] = 0;
373     state[11] = 0;
374     break;
375 }
376 case 'n': {
377
378     state[0] = 1; //1st servo
379     state[1] = 0;
380     state[2] = 0;
381
382     state[3] = 0; //2nd servo
383     state[4] = 0;
384     state[5] = 1;
385
386     state[6] = 0; //3rd servo
387     state[7] = 1;
388     state[8] = 1 ;
389
390     state[9] = 0; //4th servo
391     state[10] = 0;
392     state[11] = 0;
393     break;
394 }
395 case 'o': {
396
397     state[0] = 1; //1st servo
398     state[1] = 0;
399     state[2] = 0;
400
401     state[3] = 0; //2nd servo
402     state[4] = 0;
403     state[5] = 1;
404
405     state[6] = 0; //3rd servo
406     state[7] = 0;
407     state[8] = 1 ;
408
409     state[9] = 0; //4th servo
410     state[10] = 0;
411     state[11] = 0;
412     break;
413 }
414 case 'p': {
415
416     state[0] = 0; //1st servo
417     state[1] = 1;
418     state[2] = 0;
419
420     state[3] = 0; //2nd servo
421     state[4] = 0;
422     state[5] = 1;
423
424     state[6] = 1; //3rd servo
425     state[7] = 0;
426     state[8] = 0 ;
427
428     state[9] = 0; //4th servo
429     state[10] = 0;
430     state[11] = 0;
431     break;
432 }
433 case 'q': {
434
435     state[0] = 0; //1st servo
436     state[1] = 1;
437     state[2] = 0;
438
439     state[3] = 0; //2nd servo
440     state[4] = 0;
441     state[5] = 1;
442
443     state[6] = 0; //3rd servo
444     state[7] = 1;
445     state[8] = 0 ;
446
447     state[9] = 0; //4th servo
448     state[10] = 0;
449     state[11] = 0;
450     break;
451 }
452 case 'r': {
453
454     state[0] = 0; //1st servo
455     state[1] = 1;
456     state[2] = 0;
457
458     state[3] = 0; //2nd servo
459     state[4] = 0;
460     state[5] = 1;
461
462     state[6] = 0; //3rd servo
463     state[7] = 0;
464     state[8] = 1 ;
465
466     state[9] = 0; //4th servo
467     state[10] = 0;
468     state[11] = 0;
469     break;
470 }
471 case 's': {
472
473     state[0] = 0; //1st servo
474     state[1] = 0;
475     state[2] = 1;
476
477     state[3] = 0; //2nd servo
478     state[4] = 0;
479     state[5] = 1;
480

```

```

481     state[6] = 1; //3rd servo
482     state[7] = 0;
483     state[8] = 0 ;
484
485     state[9] = 0; //4th servo
486     state[10] = 0;
487     state[11] = 0;
488     break;
489 }
490 case 't': {
491     state[0] = 0; //1st servo
492     state[1] = 0;
493     state[2] = 1;
494
495     state[3] = 0; //2nd servo
496     state[4] = 0;
497     state[5] = 1;
498
499     state[6] = 0; //3rd servo
500     state[7] = 1;
501     state[8] = 0 ;
502
503     state[9] = 0; //4th servo
504     state[10] = 0;
505     state[11] = 0;
506     break;
507 }
508 case 'u': {
509     state[0] = 1; //1st servo
510     state[1] = 0;
511     state[2] = 0;
512
513     state[3] = 0; //2nd servo
514     state[4] = 0;
515     state[5] = 1;
516
517     state[6] = 0; //3rd servo
518     state[7] = 0;
519     state[8] = 0 ;
520
521     state[9] = 0; //4th servo
522     state[10] = 0;
523     state[11] = 1;
524     break;
525 }
526 case 'v': {
527     state[0] = 0; //1st servo
528     state[1] = 1;
529     state[2] = 0;
530
531     state[3] = 0; //2nd servo
532     state[4] = 0;
533     state[5] = 1;
534
535     state[6] = 0; //3rd servo
536     state[7] = 0;
537     state[8] = 0 ;
538
539     state[9] = 0; //4th servo
540     state[10] = 0;
541     state[11] = 1;
542     break;
543 }
544 case 'w': {
545     state[0] = 0; //1st servo
546     state[1] = 0;
547     state[2] = 1;
548
549     state[3] = 0; //2nd servo
550     state[4] = 0;
551     state[5] = 0;
552
553     state[6] = 0; //3rd servo
554     state[7] = 1;
555     state[8] = 0 ;
556
557     state[9] = 0; //4th servo
558     state[10] = 0;
559     state[11] = 3;
560
561     }
562 case 'x': {
563     state[0] = 1; //1st servo
564     state[1] = 0;
565     state[2] = 0;
566
567     state[3] = 0; //2nd servo
568     state[4] = 0;
569     state[5] = 1;
570
571     state[6] = 1; //3rd servo
572     state[7] = 0;
573     state[8] = 0 ;
574
575     state[9] = 0; //4th servo
576     state[10] = 0;
577     state[11] = 1;
578     break;
579 }
580 case 'y': {
581     state[0] = 1; //1st servo
582     state[1] = 0;
583     state[2] = 0;
584
585     state[3] = 0; //2nd servo
586     state[4] = 0;
587     state[5] = 1;
588
589     state[6] = 0; //3rd servo
590     state[7] = 1;
591     state[8] = 0 ;
592
593     state[9] = 0; //4th servo
594     state[10] = 0;
595     state[11] = 1;
596     break;
597 }
598 case 'z': {
599     state[0] = 1; //1st servo
600     state[1] = 0;
601     state[2] = 0;
602
603     state[3] = 0; //2nd servo
604     state[4] = 0;
605     state[5] = 1;
606
607     state[6] = 0; //3rd servo
608     state[7] = 0;
609     state[8] = 1 ;
610
611     state[9] = 0; //4th servo
612     state[10] = 0;
613     state[11] = 1;
614     break;
615 }
616 // CAPITAL LETTERS
617
618 case 'A': {
619     state[0] = 1; //1st servo
620     state[1] = 0;
621     state[2] = 0;
622
623     state[3] = 1; //2nd servo
624     state[4] = 0;
625     state[5] = 0;
626
627     state[6] = 0; //3rd servo
628     state[7] = 0;
629     state[8] = 0;
630
631     state[9] = 0; //4th servo
632     state[10] = 0;
633     state[11] = 0;
634     break;
635 }
636 case 'B': {
637     state[0] = 0; //1st servo
638     state[1] = 1;
639     state[2] = 0;
640

```

```

641     state[3] = 1; //2nd servo
642     state[4] = 0;
643     state[5] = 0;
644
645     state[6] = 0; //3rd servo
646     state[7] = 0;
647     state[8] = 0;
648
649     state[9] = 0; //4th servo
650     state[10] = 0;
651     state[11] = 0;
652     break;
653 }
654 case 'C': {
655     state[0] = 1; //1st servo
656     state[1] = 0;
657     state[2] = 0;
658
659     state[3] = 1; //2nd servo
660     state[4] = 0;
661     state[5] = 0;
662
663     state[6] = 1; //3rd servo
664     state[7] = 0;
665     state[8] = 0;
666
667     state[9] = 0; //4th servo
668     state[10] = 0;
669     state[11] = 0;
670     break;
671 }
672 case 'D': {
673     state[0] = 1; //1st servo
674     state[1] = 0;
675     state[2] = 0;
676
677     state[3] = 1; //2nd servo
678     state[4] = 0;
679     state[5] = 0;
680
681     state[6] = 0; //3rd servo
682     state[7] = 1;
683     state[8] = 0;
684
685     state[9] = 0; //4th servo
686     state[10] = 0;
687     state[11] = 0;
688     break;
689 }
690 case 'E': {
691     state[0] = 1; //1st servo
692     state[1] = 0;
693     state[2] = 0;
694
695     state[3] = 1; //2nd servo
696     state[4] = 0;
697     state[5] = 0;
698
699     state[6] = 0; //3rd servo
700     state[7] = 0;
701     state[8] = 1 ;
702
703     state[9] = 0; //4th servo
704     state[10] = 0;
705     state[11] = 0;
706     break;
707 }
708 case 'F': {
709     state[0] = 0; //1st servo
710     state[1] = 1;
711     state[2] = 0;
712
713     state[3] = 1; //2nd servo
714     state[4] = 0;
715     state[5] = 0;
716
717     state[6] = 1; //3rd servo
718     state[7] = 0;
719     state[8] = 0 ;
720
721     state[9] = 0; //4th servo
722     state[10] = 0;
723     state[11] = 0;
724     break;
725 }
726 case 'G': {
727     state[0] = 0; //1st servo
728     state[1] = 1;
729     state[2] = 0;
730
731     state[3] = 1; //2nd servo
732     state[4] = 0;
733     state[5] = 0;
734
735     state[6] = 0; //3rd servo
736     state[7] = 1;
737     state[8] = 0 ;
738
739     state[9] = 0; //4th servo
740     state[10] = 0;
741     state[11] = 0;
742     break;
743 }
744 case 'H': {
745     state[0] = 0; //1st servo
746     state[1] = 1;
747     state[2] = 0;
748
749     state[3] = 1; //2nd servo
750     state[4] = 0;
751     state[5] = 0;
752
753     state[6] = 0; //3rd servo
754     state[7] = 0;
755     state[8] = 1 ;
756
757     state[9] = 0; //4th servo
758     state[10] = 0;
759     state[11] = 0;
760     break;
761 }
762 case 'I': {
763     state[0] = 0; //1st servo
764     state[1] = 0;
765     state[2] = 1;
766
767     state[3] = 1; //2nd servo
768     state[4] = 0;
769     state[5] = 0;
770
771     state[6] = 1; //3rd servo
772     state[7] = 0;
773     state[8] = 0 ;
774
775     state[9] = 0; //4th servo
776     state[10] = 0;
777     state[11] = 0;
778     break;
779 }
780 case 'J': {
781     state[0] = 0; //1st servo
782     state[1] = 0;
783     state[2] = 1;
784
785     state[3] = 1; //2nd servo
786     state[4] = 0;
787     state[5] = 0;
788
789     state[6] = 0; //3rd servo
790     state[7] = 1;
791     state[8] = 0 ;
792
793     state[9] = 0; //4th servo
794     state[10] = 0;
795     state[11] = 0;
796     break;
797 }
798 case 'K': {
799     state[0] = 1; //1st servo
800     state[1] = 0;

```

```

801     state[2] = 0;
802
803
804
805     state[3] = 0; //2nd servo
806     state[4] = 1;
807     state[5] = 0;
808
809
810
811     state[6] = 0; //3rd servo
812     state[7] = 0;
813     state[8] = 0 ;
814
815
816
817     state[9] = 0; //4th servo
818     state[10] = 0;
819     state[11] = 0;
820     break;
821 }
822 case 'L': {
823     state[0] = 0; //1st servo
824     state[1] = 1;
825     state[2] = 0;
826
827     state[3] = 0; //2nd servo
828     state[4] = 1;
829     state[5] = 0;
830
831     state[6] = 0; //3rd servo
832     state[7] = 0;
833     state[8] = 0 ;
834
835     state[9] = 0; //4th servo
836     state[10] = 0;
837     state[11] = 0;
838     break;
839 }
840 case 'M': {
841     state[0] = 1; //1st servo
842     state[1] = 0;
843     state[2] = 0;
844
845     state[3] = 0; //2nd servo
846     state[4] = 1;
847     state[5] = 0;
848
849     state[6] = 1; //3rd servo
850     state[7] = 0;
851     state[8] = 0 ;
852
853     state[9] = 0; //4th servo
854     state[10] = 0;
855     state[11] = 0;
856     break;
857 }
858 case 'N': {
859     state[0] = 1; //1st servo
860     state[1] = 0;
861     state[2] = 0;
862
863     state[3] = 0; //2nd servo
864     state[4] = 1;
865     state[5] = 0;
866
867     state[6] = 0; //3rd servo
868     state[7] = 1;
869     state[8] = 0;
870
871     state[9] = 0; //4th servo
872     state[10] = 0;
873     state[11] = 0;
874     break;
875 }
876 case 'O': {
877     state[0] = 1; //1st servo
878     state[1] = 0;
879     state[2] = 0;
880
881     state[3] = 0; //2nd servo
882     state[4] = 1;
883     state[5] = 0;
884
885     state[6] = 0; //3rd servo
886     state[7] = 0;
887     state[8] = 1 ;
888
889     state[9] = 0; //4th servo
890     state[10] = 0;
891     state[11] = 0;
892     break;
893 }
894 case 'P': {
895     state[0] = 0; //1st servo
896     state[1] = 1;
897     state[2] = 0;
898
899     state[3] = 0; //2nd servo
900     state[4] = 1;
901     state[5] = 0;
902
903     state[6] = 1; //3rd servo
904     state[7] = 0;
905     state[8] = 0 ;
906
907     state[9] = 0; //4th servo
908     state[10] = 0;
909     state[11] = 0;
910     break;
911 }
912 case 'Q': {
913     state[0] = 0; //1st servo
914     state[1] = 1;
915     state[2] = 0;
916
917     state[3] = 0; //2nd servo
918     state[4] = 1;
919     state[5] = 0;
920
921     state[6] = 0; //3rd servo
922     state[7] = 1;
923     state[8] = 0 ;
924
925     state[9] = 0; //4th servo
926     state[10] = 0;
927     state[11] = 0;
928     break;
929 }
930 case 'R': {
931     state[0] = 0; //1st servo
932     state[1] = 1;
933     state[2] = 0;
934
935     state[3] = 0; //2nd servo
936     state[4] = 1;
937     state[5] = 0;
938
939     state[6] = 0; //3rd servo
940     state[7] = 0;
941     state[8] = 1 ;
942
943     state[9] = 0; //4th servo
944     state[10] = 0;
945     state[11] = 0;
946     break;
947 }
948 case 'S': {
949     state[0] = 0; //1st servo
950     state[1] = 0;
951     state[2] = 1;
952
953     state[3] = 0; //2nd servo
954     state[4] = 1;
955     state[5] = 0;
956
957     state[6] = 1; //3rd servo
958     state[7] = 0;
959     state[8] = 0 ;
960

```

```

961     state[9] = 0; //4th servo
962     state[10] = 0;
963     state[11] = 0;
964     break;
965 }
966 case 'T': {
967     state[0] = 0; //1st servo
968     state[1] = 0;
969     state[2] = 1;
970
971     state[3] = 0; //2nd servo
972     state[4] = 1;
973     state[5] = 0;
974
975     state[6] = 0; //3rd servo
976     state[7] = 1;
977     state[8] = 0 ;
978
979     state[9] = 0; //4th servo
980     state[10] = 0;
981     state[11] = 0;
982     break;
983 }
984 case 'U': {
985     state[0] = 1; //1st servo
986     state[1] = 0;
987     state[2] = 0;
988
989     state[3] = 0; //2nd servo
990     state[4] = 1;
991     state[5] = 0;
992
993     state[6] = 0; //3rd servo
994     state[7] = 0;
995     state[8] = 0 ;
996
997     state[9] = 0; //4th servo
998     state[10] = 0;
999     state[11] = 1;
1000     break;
1001 }
1002 case 'V': {
1003     state[0] = 0; //1st servo
1004     state[1] = 1;
1005     state[2] = 0;
1006
1007     state[3] = 0; //2nd servo
1008     state[4] = 1;
1009     state[5] = 0;
1010
1011     state[6] = 0; //3rd servo
1012     state[7] = 0;
1013     state[8] = 0 ;
1014
1015     state[9] = 0; //4th servo
1016     state[10] = 0;
1017     state[11] = 1;
1018     break;
1019 }
1020 case 'W': {
1021     state[0] = 0; //1st servo
1022     state[1] = 0;
1023     state[2] = 1;
1024
1025     state[3] = 1; //2nd servo
1026     state[4] = 0;
1027     state[5] = 0;
1028
1029     state[6] = 0; //3rd servo
1030     state[7] = 1;
1031     state[8] = 0 ;
1032
1033     state[9] = 0; //4th servo
1034     state[10] = 0;
1035     state[11] = 1;
1036     break;
1037 }
1038 case 'X': {
1039     state[0] = 1; //1st servo
1040     state[1] = 0;
1041     state[2] = 0;
1042
1043     state[3] = 0; //2nd servo
1044     state[4] = 1;
1045     state[5] = 0;
1046
1047     state[6] = 1; //3rd servo
1048     state[7] = 0;
1049     state[8] = 0 ;
1050
1051     state[9] = 0; //4th servo
1052     state[10] = 0;
1053     state[11] = 1;
1054     break;
1055 }
1056 case 'Y': {
1057     state[0] = 1; //1st servo
1058     state[1] = 0;
1059     state[2] = 0;
1060
1061     state[3] = 0; //2nd servo
1062     state[4] = 1;
1063     state[5] = 0;
1064
1065     state[6] = 0; //3rd servo
1066     state[7] = 1;
1067     state[8] = 0 ;
1068
1069     state[9] = 0; //4th servo
1070     state[10] = 0;
1071     state[11] = 1;
1072     break;
1073 }
1074 case 'Z': {
1075     state[0] = 1; //1st servo
1076     state[1] = 0;
1077     state[2] = 0;
1078
1079     state[3] = 0; //2nd servo
1080     state[4] = 1;
1081     state[5] = 0;
1082
1083     state[6] = 0; //3rd servo
1084     state[7] = 0;
1085     state[8] = 1 ;
1086
1087     state[9] = 0; //4th servo
1088     state[10] = 0;
1089     state[11] = 1;
1090     break;
1091 }
1092 //NUMBERS
1093 case '0': {
1094     state[0] = 0; //1st servo
1095     state[1] = 0;
1096     state[2] = 0;
1097
1098     state[3] = 0; //2nd servo
1099     state[4] = 0;
1100     state[5] = 1;
1101
1102     state[6] = 0; //3rd servo
1103     state[7] = 1;
1104     state[8] = 0 ;
1105
1106     state[9] = 0; //4th servo
1107     state[10] = 0;
1108     state[11] = 1;
1109     break;
1110 }
1111 case '1': {
1112     state[0] = 1; //1st servo
1113     state[1] = 0;
1114     state[2] = 0;
1115
1116     state[3] = 0; //2nd servo
1117     state[4] = 0;
1118     state[5] = 0;
1119
1120     state[6] = 0; //3rd servo

```

```

1121     state[7] = 0;
1122     state[8] = 0 ;
1123
1124     state[9] = 0; //4th servo
1125     state[10] = 0;
1126     state[11] = 1;
1127     break;
1128 }
1129 case '2': {
1130     state[0] = 0; //1st servo
1131     state[1] = 1;
1132     state[2] = 0;
1133
1134     state[3] = 0; //2nd servo
1135     state[4] = 0;
1136     state[5] = 0;
1137
1138     state[6] = 0; //3rd servo
1139     state[7] = 0;
1140     state[8] = 0 ;
1141
1142     state[9] = 0; //4th servo
1143     state[10] = 0;
1144     state[11] = 1;
1145     break;
1146 }
1147 case '3': {
1148     state[0] = 1; //1st servo
1149     state[1] = 0;
1150     state[2] = 0;
1151
1152     state[3] = 0; //2nd servo
1153     state[4] = 0;
1154     state[5] = 0;
1155
1156     state[6] = 1; //3rd servo
1157     state[7] = 0;
1158     state[8] = 0 ;
1159
1160     state[9] = 0; //4th servo
1161     state[10] = 0;
1162     state[11] = 1;
1163     break;
1164 }
1165 case '4': {
1166     state[0] = 1; //1st servo
1167     state[1] = 0;
1168     state[2] = 0;
1169
1170     state[3] = 0; //2nd servo
1171     state[4] = 0;
1172     state[5] = 0;
1173
1174     state[6] = 0; //3rd servo
1175     state[7] = 1;
1176     state[8] = 0 ;
1177
1178     state[9] = 0; //4th servo
1179     state[10] = 0;
1180     state[11] = 1;
1181     break;
1182 }
1183 case '5': {
1184     state[0] = 1; //1st servo
1185     state[1] = 0;
1186     state[2] = 0;
1187
1188     state[3] = 0; //2nd servo
1189     state[4] = 0;
1190     state[5] = 0;
1191
1192     state[6] = 0; //3rd servo
1193     state[7] = 0;
1194     state[8] = 1 ;
1195
1196     state[9] = 0; //4th servo
1197     state[10] = 0;
1198     state[11] = 1;
1199     break;
1200 }
1201 case '6': {
1202     state[0] = 0; //1st servo
1203     state[1] = 1;
1204     state[2] = 0;
1205
1206     state[3] = 0; //2nd servo
1207     state[4] = 0;
1208     state[5] = 0;
1209
1210     state[6] = 1; //3rd servo
1211     state[7] = 0;
1212     state[8] = 0 ;
1213
1214     state[9] = 0; //4th servo
1215     state[10] = 0;
1216     state[11] = 1;
1217     break;
1218 }
1219 case '7': {
1220     state[0] = 0; //1st servo
1221     state[1] = 1;
1222     state[2] = 0;
1223
1224     state[3] = 0; //2nd servo
1225     state[4] = 0;
1226     state[5] = 0;
1227
1228     state[6] = 0; //3rd servo
1229     state[7] = 1;
1230     state[8] = 0 ;
1231
1232     state[9] = 0; //4th servo
1233     state[10] = 0;
1234     state[11] = 1;
1235     break;
1236 }
1237 case '8': {
1238     state[0] = 0; //1st servo
1239     state[1] = 1;
1240     state[2] = 0;
1241
1242     state[3] = 0; //2nd servo
1243     state[4] = 0;
1244     state[5] = 0;
1245
1246     state[6] = 0; //3rd servo
1247     state[7] = 0;
1248     state[8] = 1 ;
1249
1250     state[9] = 0; //4th servo
1251     state[10] = 0;
1252     state[11] = 1;
1253
1254     break;
1255 }
1256 case '9': {
1257     state[0] = 0; //1st servo
1258     state[1] = 0;
1259     state[2] = 1;
1260
1261     state[3] = 0; //2nd servo
1262     state[4] = 0;
1263     state[5] = 0;
1264
1265     state[6] = 1; //3rd servo
1266     state[7] = 0;
1267     state[8] = 0;
1268
1269     state[9] = 0; //4th servo
1270     state[10] = 0;
1271     state[11] = 1;
1272     break;
1273 }
1274 //Punctuation
1275 case '!': {
1276     state[0] = 0; //1st servo
1277     state[1] = 0;
1278     state[2] = 1;
1279
1280     state[3] = 0; //2nd servo

```

```

1281     state[4] = 0;
1282     state[5] = 1;
1283
1284     state[6] = 0; //3rd servo
1285     state[7] = 0;
1286     state[8] = 1;
1287
1288     state[9] = 0; //4th servo
1289     state[10] = 0;
1290     state[11] = 0;
1291     break;
1292 }
1293 case '(': {
1294     state[0] = 0; //1st servo
1295     state[1] = 0;
1296     state[2] = 1;
1297
1298     state[3] = 0; //2nd servo
1299     state[4] = 0;
1300     state[5] = 1;
1301
1302     state[6] = 0; //3rd servo
1303     state[7] = 0;
1304     state[8] = 0 ;
1305
1306     state[9] = 0; //4th servo
1307     state[10] = 0;
1308     state[11] = 1;
1309     break;
1310 }
1311 case ')': {
1312     state[0] = 0; //1st servo
1313     state[1] = 0;
1314     state[2] = 0;
1315
1316     state[3] = 0; //2nd servo
1317     state[4] = 0;
1318     state[5] = 1;
1319
1320     state[6] = 0; //3rd servo
1321     state[7] = 0;
1322     state[8] = 1 ;
1323
1324     state[9] = 0; //4th servo
1325     state[10] = 0;
1326     state[11] = 1;
1327     break;
1328 }
1329 case '*': {
1330     state[0] = 0; //1st servo
1331     state[1] = 0;
1332     state[2] = 0;
1333
1334     state[3] = 0; //2nd servo
1335     state[4] = 0;
1336     state[5] = 1;
1337
1338     state[6] = 0; //3rd servo
1339     state[7] = 0;
1340     state[8] = 1;
1341
1342     state[9] = 0; //4th servo
1343     state[10] = 0;
1344     state[11] = 0;
1345     break;
1346 }
1347 case ',': {
1348     state[0] = 0; //1st servo
1349     state[1] = 0;
1350     state[2] = 1;
1351
1352     state[3] = 0; //2nd servo
1353     state[4] = 0;
1354     state[5] = 0;
1355
1356     state[6] = 0; //3rd servo
1357     state[7] = 0;
1358     state[8] = 0 ;
1359
1360     state[9] = 0; //4th servo
1361     state[10] = 0;
1362     state[11] = 0;
1363     break;
1364 }
1365 case '-': {
1366     state[0] = 0; //1st servo
1367     state[1] = 0;
1368     state[2] = 0;
1369
1370     state[3] = 0; //2nd servo
1371     state[4] = 0;
1372     state[5] = 1;
1373
1374     state[6] = 0; //3rd servo
1375     state[7] = 0;
1376     state[8] = 0 ;
1377
1378     state[9] = 0; //4th servo
1379     state[10] = 0;
1380     state[11] = 1;
1381     break;
1382 }
1383 case '.': {
1384     state[0] = 0; //1st servo
1385     state[1] = 0;
1386     state[2] = 0;
1387
1388     state[3] = 0; //2nd servo
1389     state[4] = 0;
1390     state[5] = 1;
1391
1392     state[6] = 0; //3rd servo
1393     state[7] = 0;
1394     state[8] = 0 ;
1395
1396     state[9] = 0; //4th servo
1397     state[10] = 0;
1398     state[11] = 0;
1399     break;
1400 }
1401 case ':': {
1402     state[0] = 0; //1st servo
1403     state[1] = 0;
1404     state[2] = 1;
1405
1406     state[3] = 0; //2nd servo
1407     state[4] = 0;
1408     state[5] = 0;
1409
1410     state[6] = 0; //3rd servo
1411     state[7] = 0;
1412     state[8] = 1 ;
1413
1414     state[9] = 0; //4th servo4
1415     state[10] = 0;
1416     state[11] = 0;
1417     break;
1418 }
1419 case ';': {
1420     state[0] = 0; //1st servo
1421     state[1] = 0;
1422     state[2] = 1;
1423
1424     state[3] = 0; //2nd servo
1425     state[4] = 0;
1426     state[5] = 1;
1427
1428     state[6] = 0; //3rd servo
1429     state[7] = 0;
1430     state[8] = 0 ;
1431
1432     state[9] = 0; //4th servo
1433     state[10] = 0;
1434     state[11] = 0;
1435     break;
1436 }
1437 case '?': {
1438     state[0] = 0; //1st servo
1439     state[1] = 0;
1440     state[2] = 1;

```

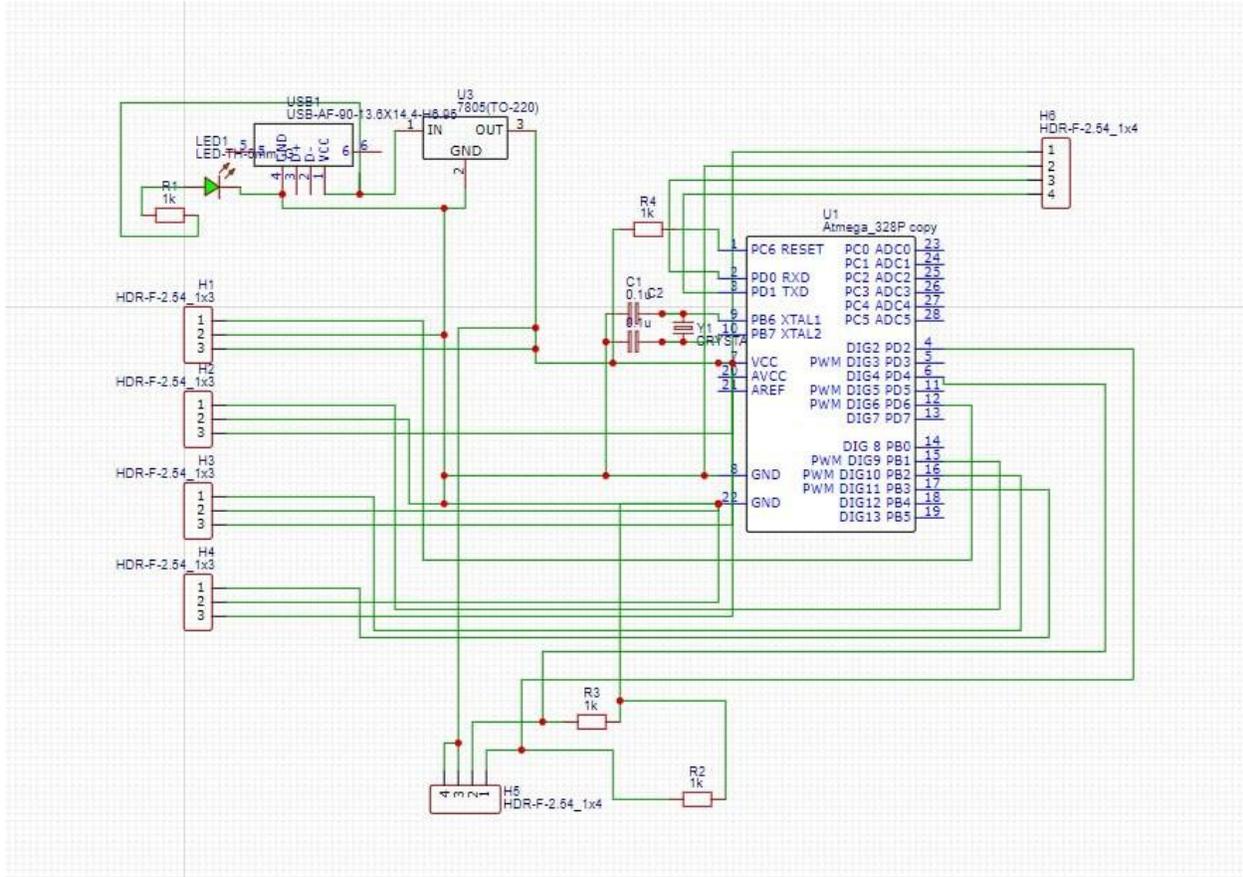
```

1441
1442     state[3] = 0; //2nd servo
1443     state[4] = 0;
1444     state[5] = 0;
1445
1446     state[6] = 0; //3rd servo
1447     state[7] = 0;
1448     state[8] = 0 ;
1449
1450     state[9] = 0; //4th servo
1451     state[10] = 0;
1452     state[11] = 1;
1453     break;
1454 }
1455 case ' ': {
1456
1457     state[0] = 0; //1st servo
1458     state[1] = 0;
1459     state[2] = 0;
1460
1461     state[3] = 0; //2nd servo
1462     state[4] = 0;
1463     state[5] = 0;
1464
1465     state[6] = 0; //3rd servo
1466     state[7] = 0;
1467     state[8] = 0;
1468
1469     state[9] = 0; //4th servo
1470     state[10] = 0;
1471     state[11] = 0;
1472     break;
1473 }
1474 case '$': {
1475     state[0] = 0; //1st servo
1476     state[1] = 1;
1477     state[2] = 0;
1478
1479     state[3] = 0; //2nd servo
1480     state[4] = 1;
1481     state[5] = 0;
1482
1483     state[6] = 0; //3rd servo
1484     state[7] = 1;
1485     state[8] = 0;
1486
1487     state[9] = 0; //4th servo
1488     state[10] = 1;
1489     state[11] = 0;
1490     break;
1491 }
1492 }
1493 }
1494 /*Displaying character which is
1495 converted to braille system*/
1496 void displayChar() {
1497     //First Servo
1498     if (state[0] || state[1] || state[2])
1499     {
1500         //1st case
1501         if (state[0])
1502         {
1503             ser1.write(65);
1504         }
1505         //2nd case
1506         else if (state[1])
1507         {
1508             ser1.write(95);
1509         }
1510         //3rd case
1511         else if (state[2])
1512         {
1513             ser1.write(135);
1514         }
1515     } else
1516     {
1517         //4th case
1518         ser1.write(20);
1519     }
1520     //Second Servo
1521     if (state[3] || state[4] || state[5])
1522     {
1523         //1st case
1524         if (state[3])
1525         {
1526             ser2.write(5);
1527         }
1528         //2nd case
1529         else if (state[4])
1530         {
1531             ser2.write(48);
1532         }
1533         //3rd case
1534         else if (state[5])
1535         {
1536             ser2.write(85);
1537         }
1538     } else
1539     {
1540         //4th case
1541         ser2.write(170);
1542     }
1543     //Third Servo
1544     if (state[6] || state[7] || state[8])
1545     {
1546         //1st case
1547         if (state[6])
1548         {
1549             ser3.write(0);
1550         }
1551         //2nd case
1552         else if (state[7])
1553         {
1554             ser3.write(40);
1555         }
1556         //3rd case
1557         else if (state[8])
1558         {
1559             ser3.write(70);
1560         }
1561     } else
1562     {
1563         //4th case
1564         ser3.write(150);
1565     }
1566     //Fourth Servo
1567     if (state[9] || state[10] || state[11])
1568     {
1569         //1st case
1570         if (state[9])
1571         {
1572             ser4.write(35);
1573         }
1574         //2nd case
1575         else if (state[10])
1576         {
1577             ser4.write(75);
1578         }
1579         //3rd case
1580         else if (state[11])
1581         {
1582             ser4.write(115);
1583         }
1584     } else
1585     {
1586         //4th case
1587         ser4.write(0);
1588     }
1589     delay(100);
1590 }

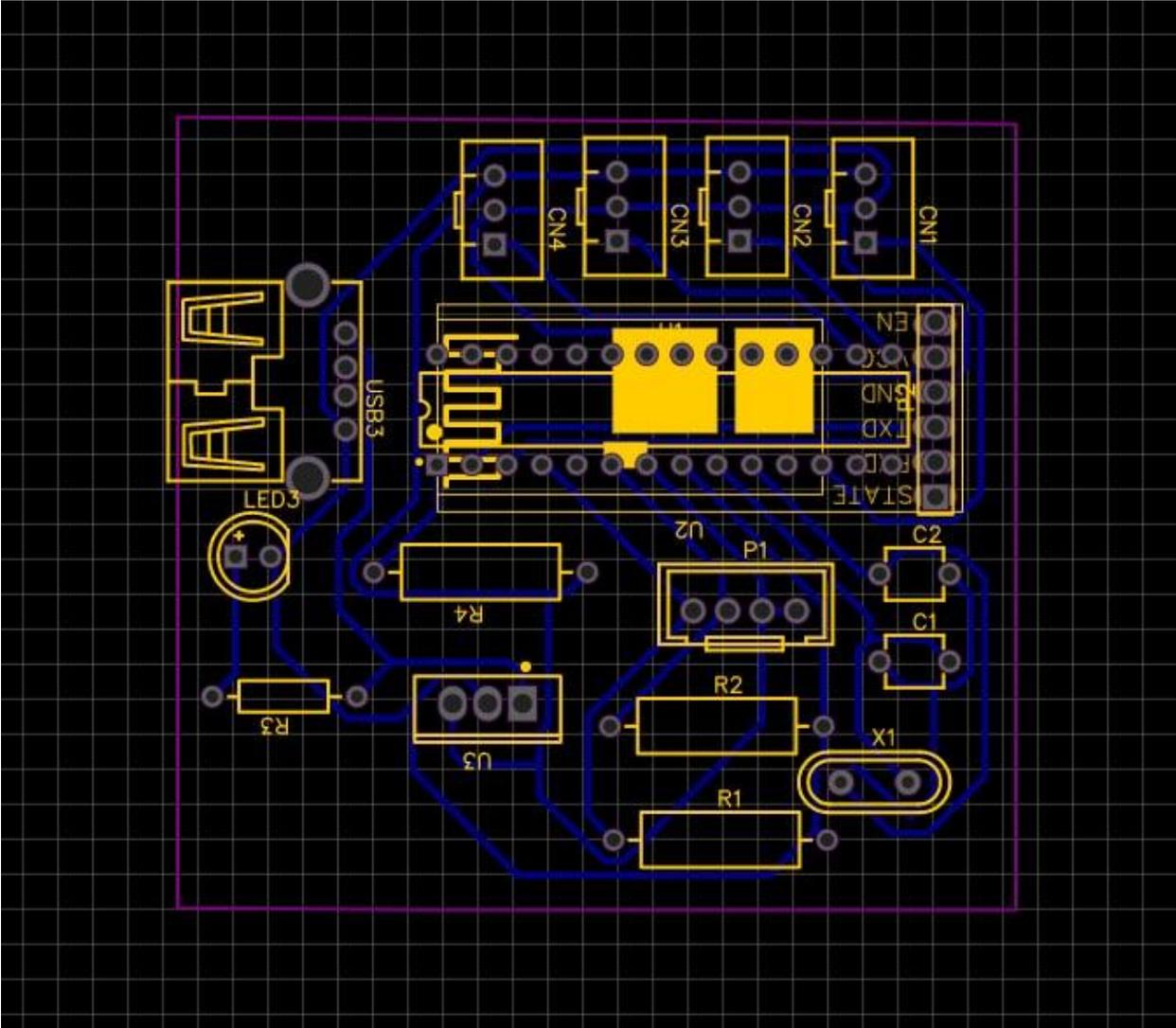
```

Appendix 6.

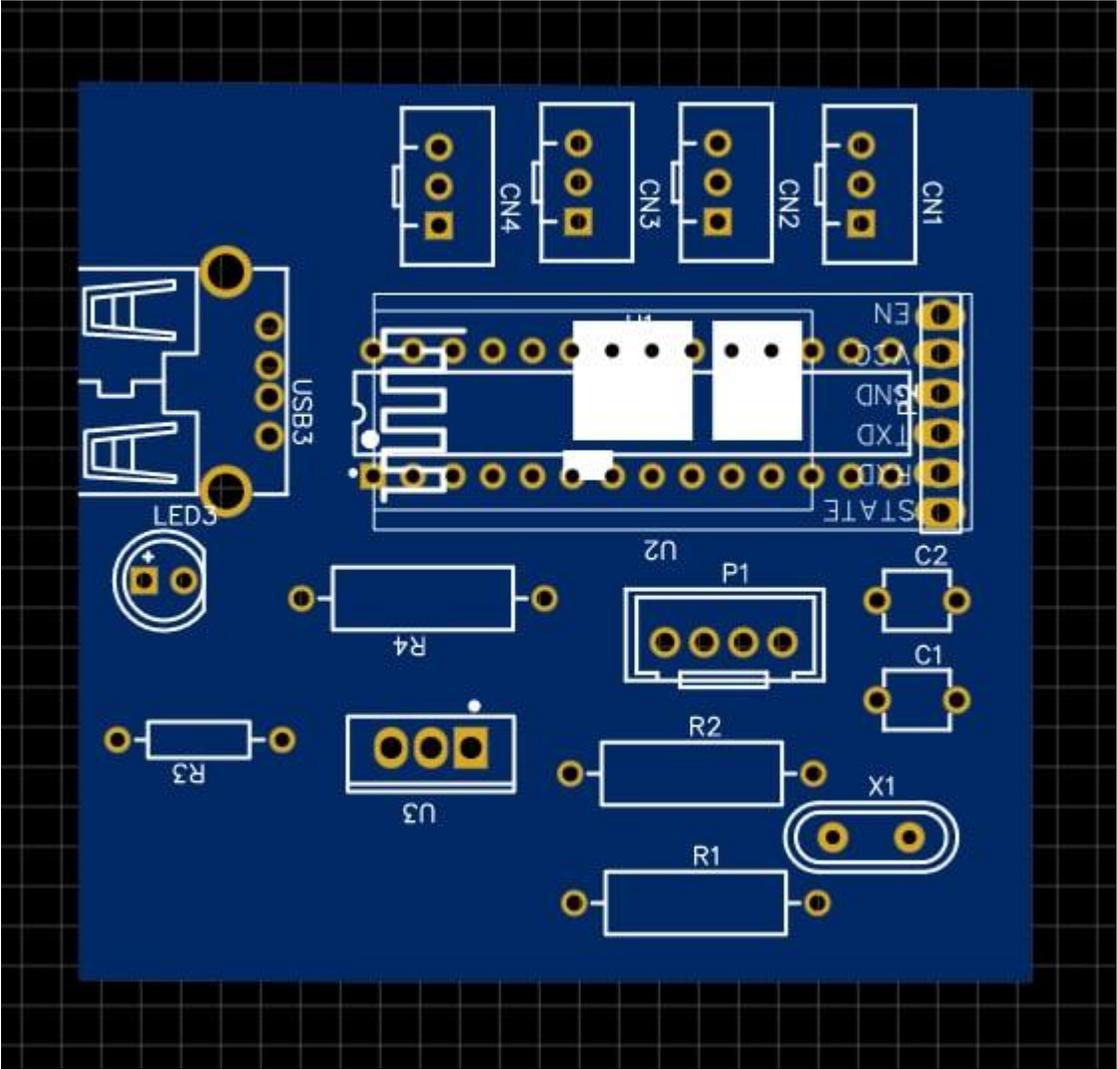
the schematics used in controlling the servo motors with Atmega328 microcontroller.



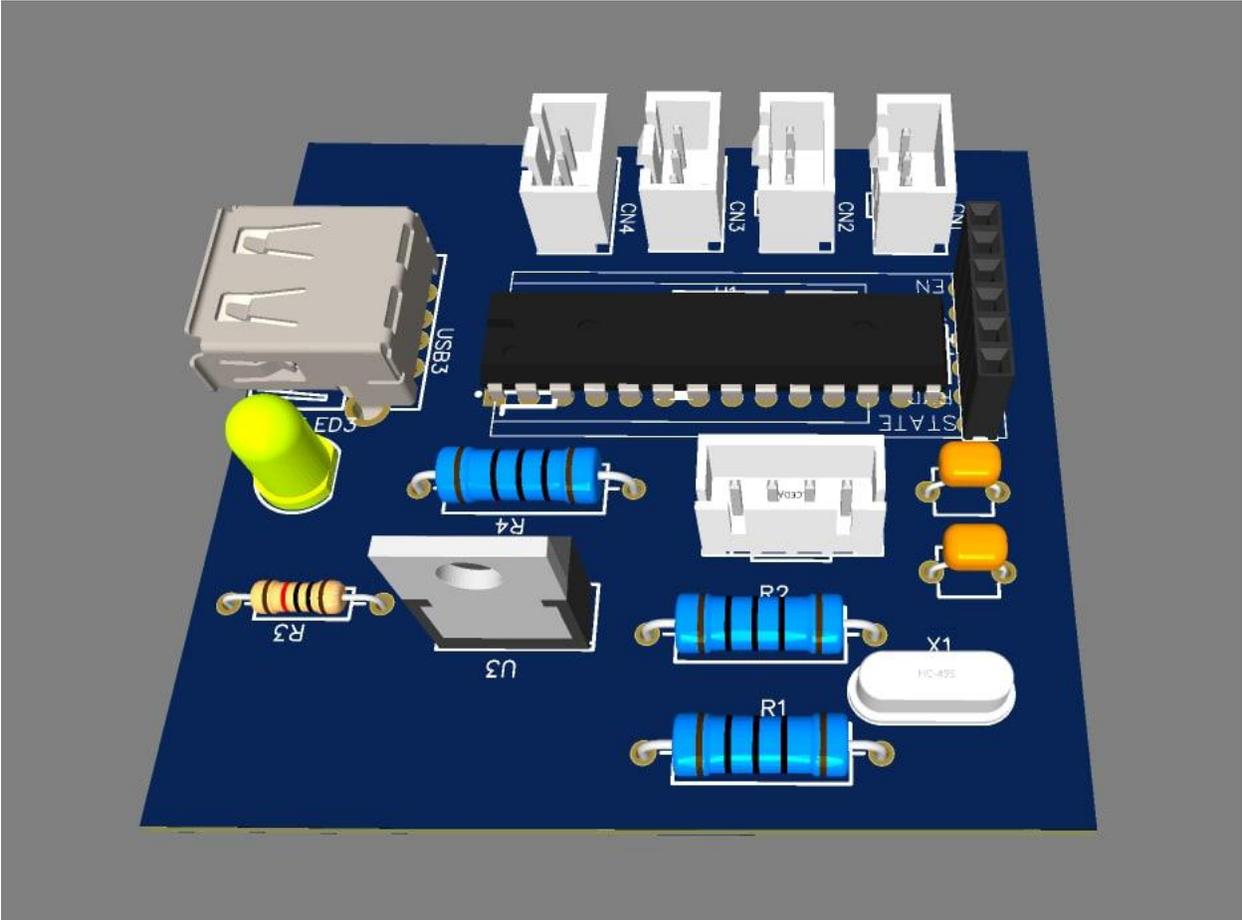
To find optimal place for Placing the components and routing lines according to main schematics



The top view of Printed Circuit Board (PCB) without components in 3D view



The top view of Printed Circuit Board (PCB) with components in 3D view



The final routed schematics for Braille Display VI version

