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.

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.

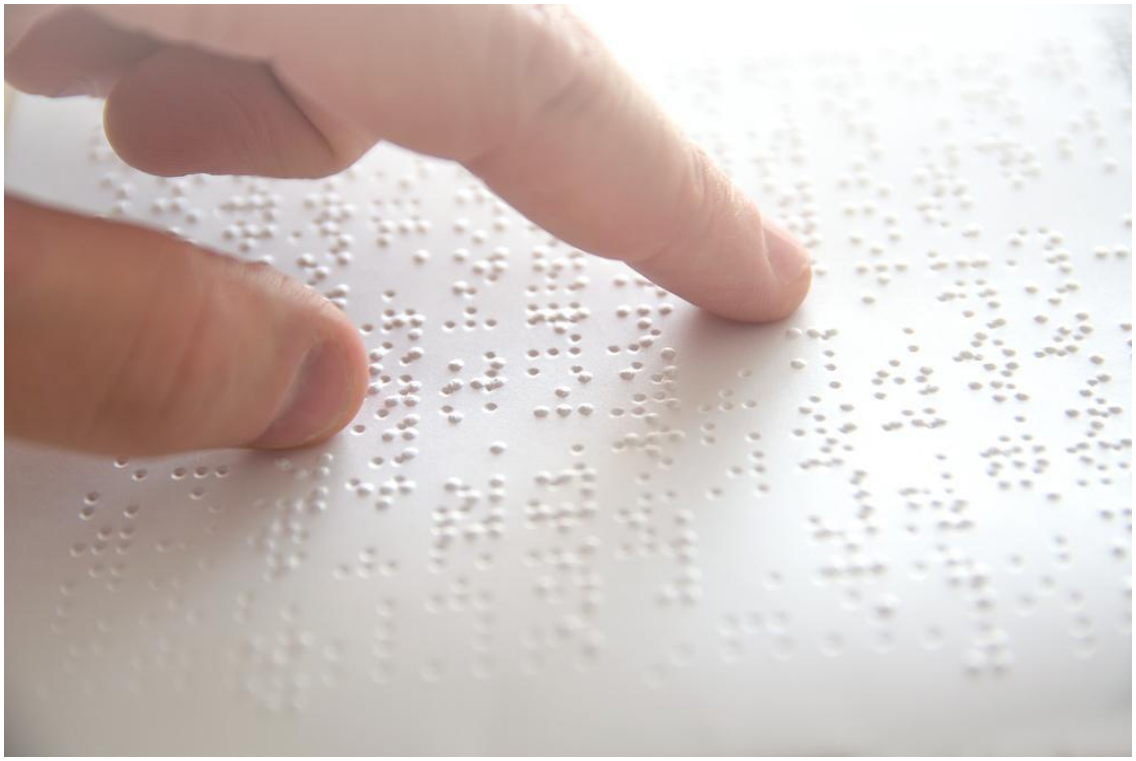


Figure 1. Braille paper [2] .

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.

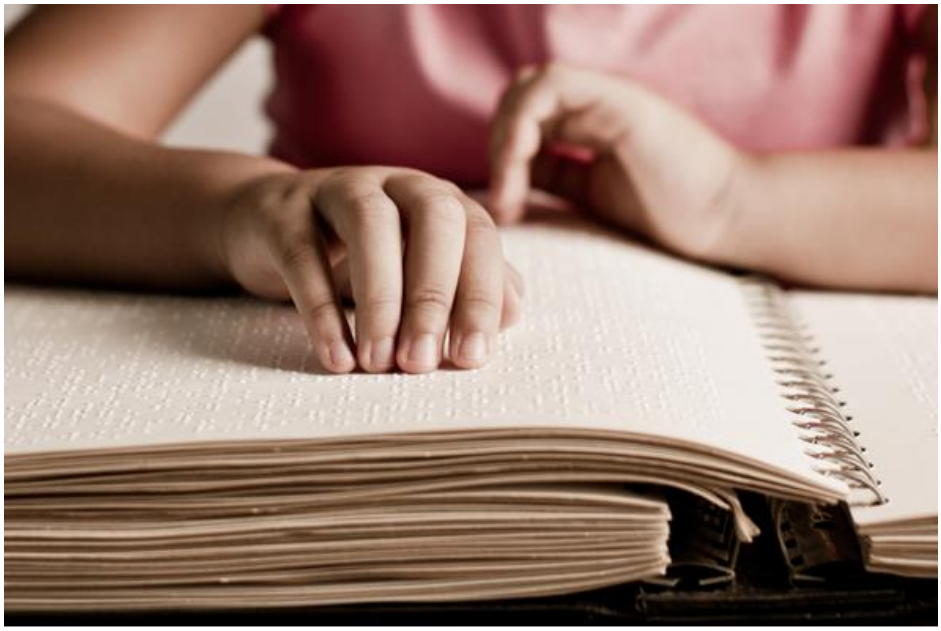


Figure 2. Braille book [3].

(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 earlies technology used by Louis Braille himself. The next generation of instruments is so called Perkins Brailler [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.

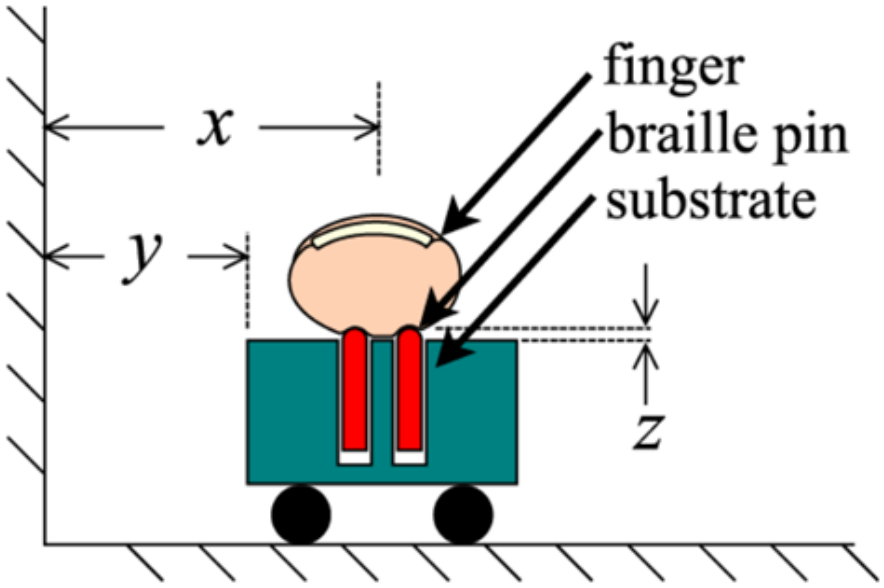


Figure 3. Finger sensing [9].

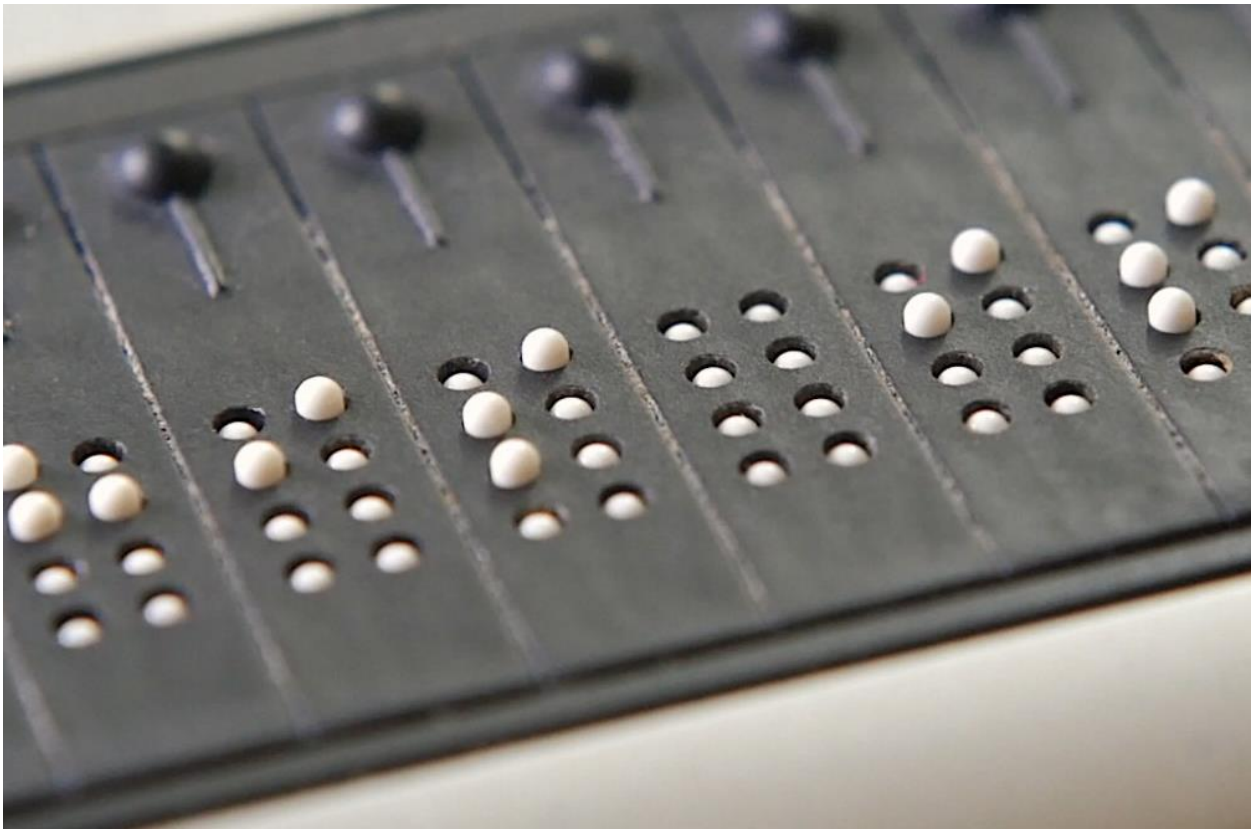


Figure 4. Dots in Braille display [10].

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.

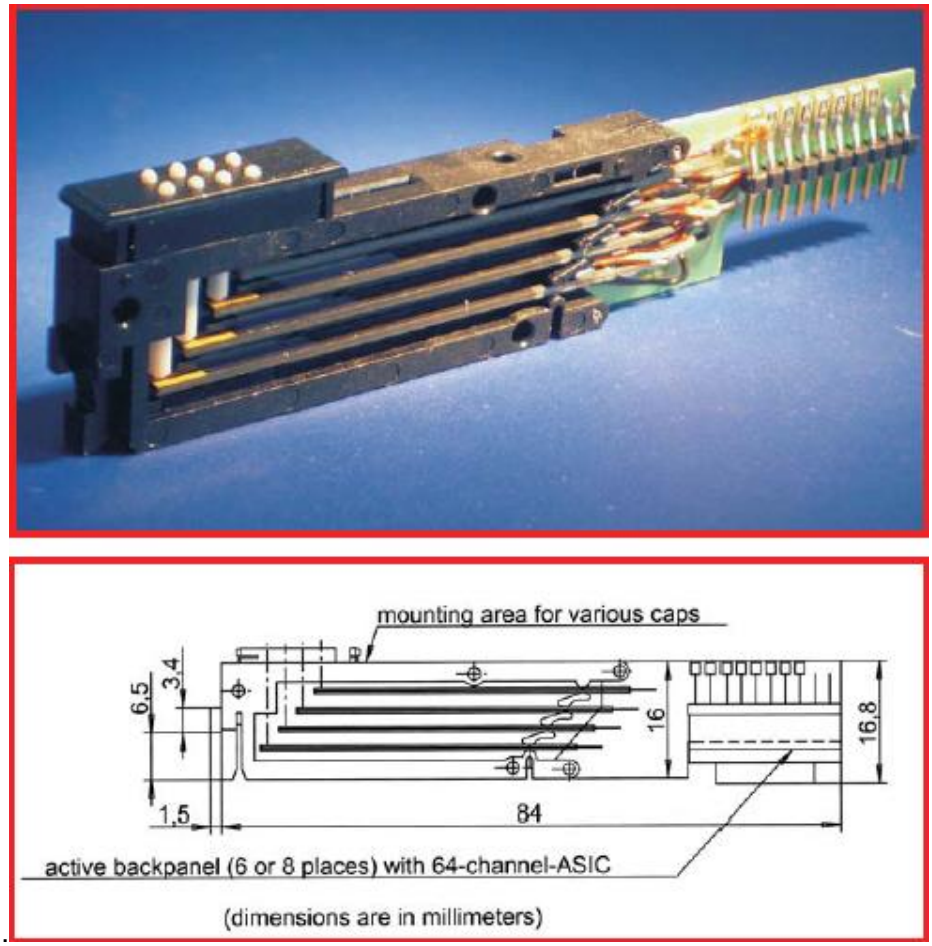


Figure 6. Piezo System in Braille display [13].

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.

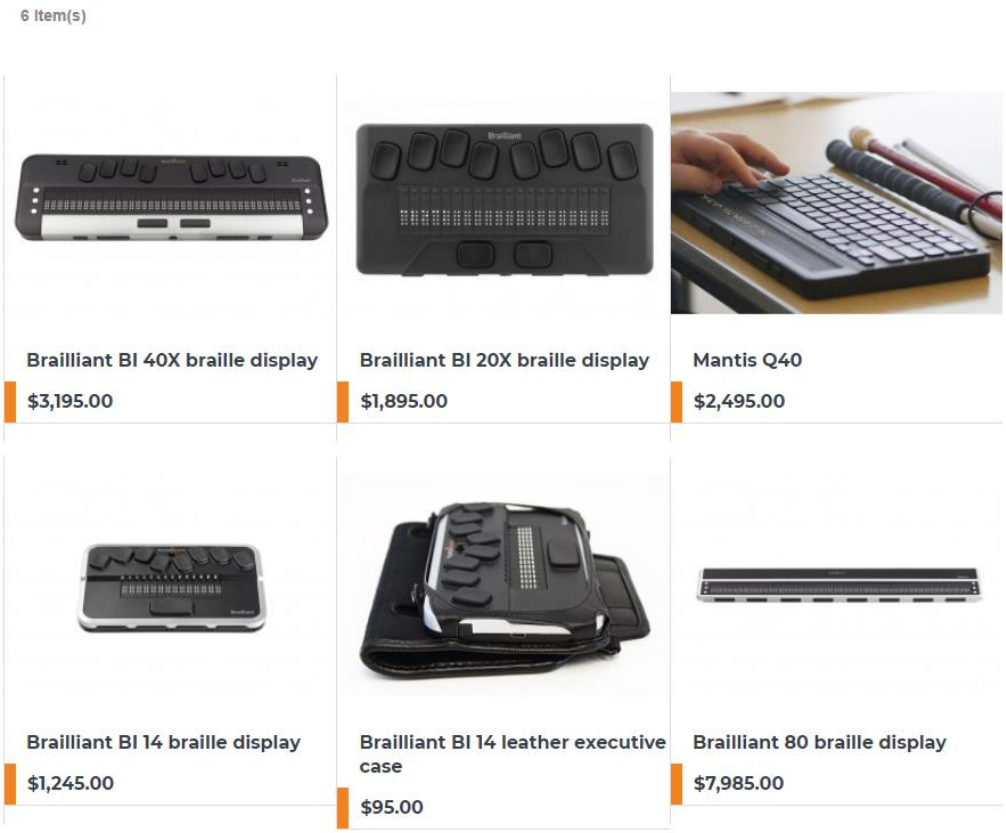


Figure 7. Braille display price range [14].

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.

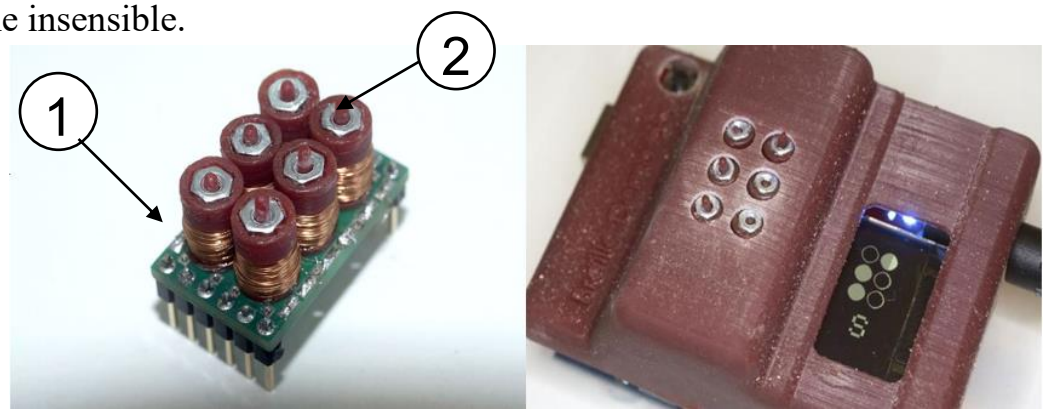
2.0 Section description.

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.



(1) Solenoid.

(2) Drive pin.

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)

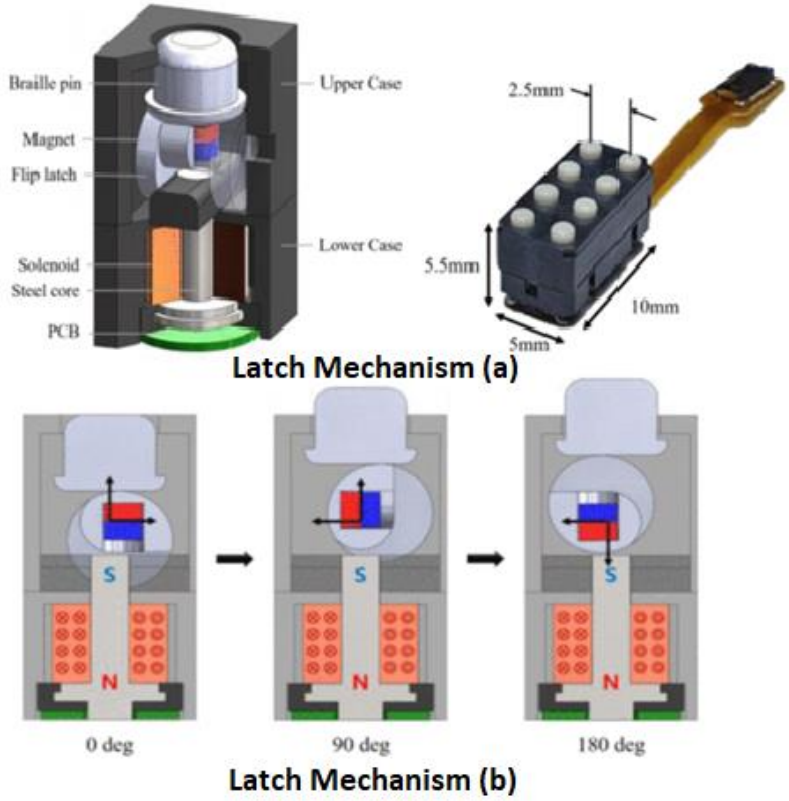


Figure 9. Latch Mechanism [19].

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.

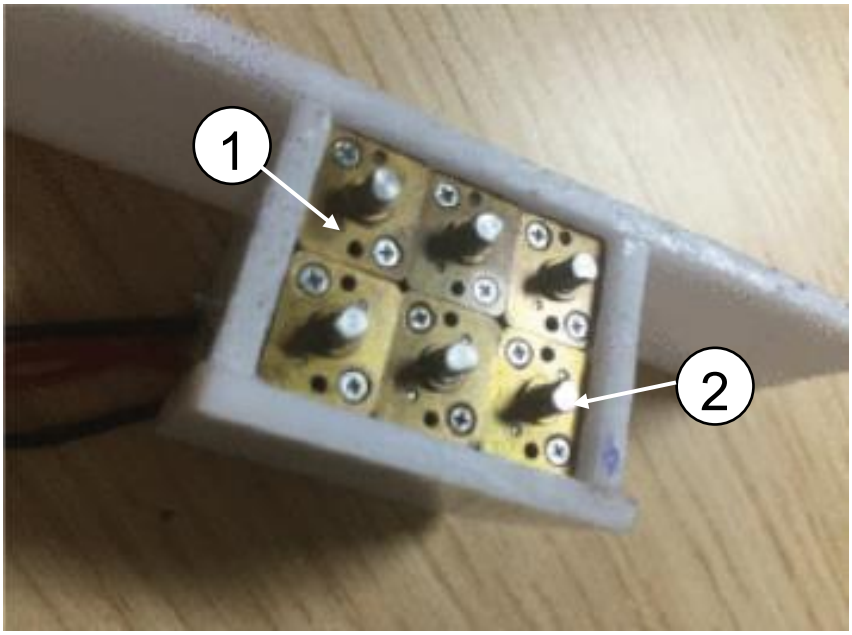
In the passive state the Northern direction of magnet that is fastened on flip latch mechanism coincides with the polarity of the electromagnet. As the electromagnet switches its polarity, the magnet turns around to fit the polarity of the

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.

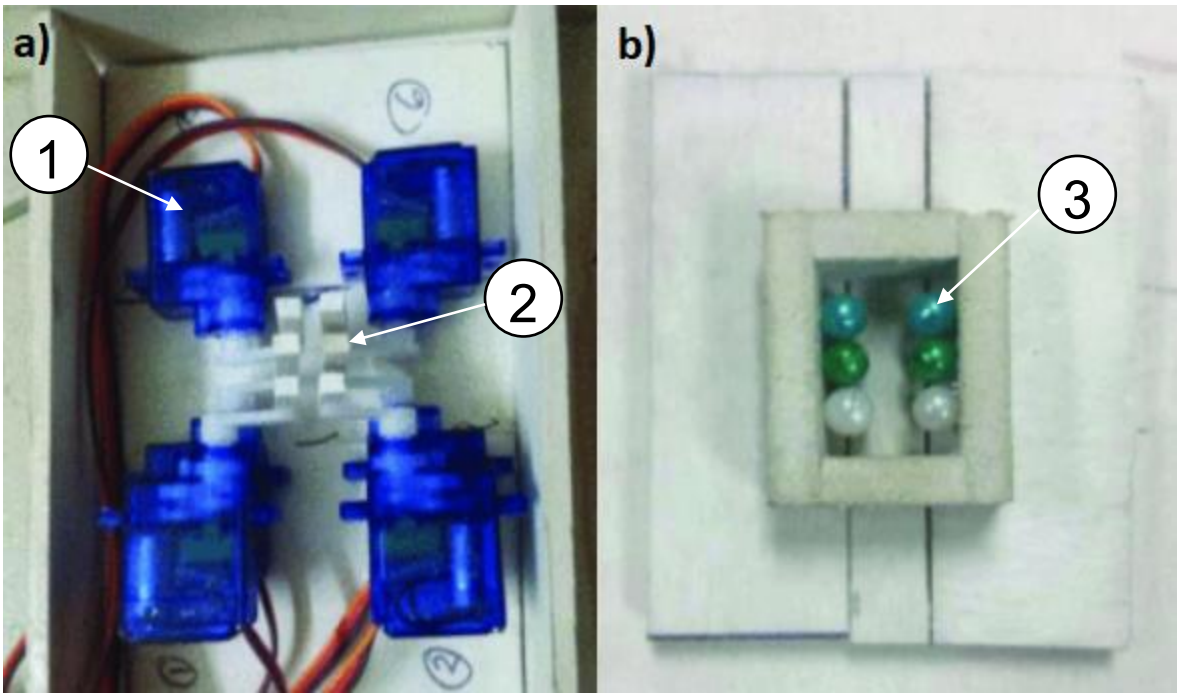
Some teams propose simple solutions, but there is little probability, that with the sizes of the actuators those researchers propose, somehow it might be possible to fit the pins to the Braille standards. Linear actuators (Figure 10) and the usage of servo motors directly for the pin (Figure 11) is one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

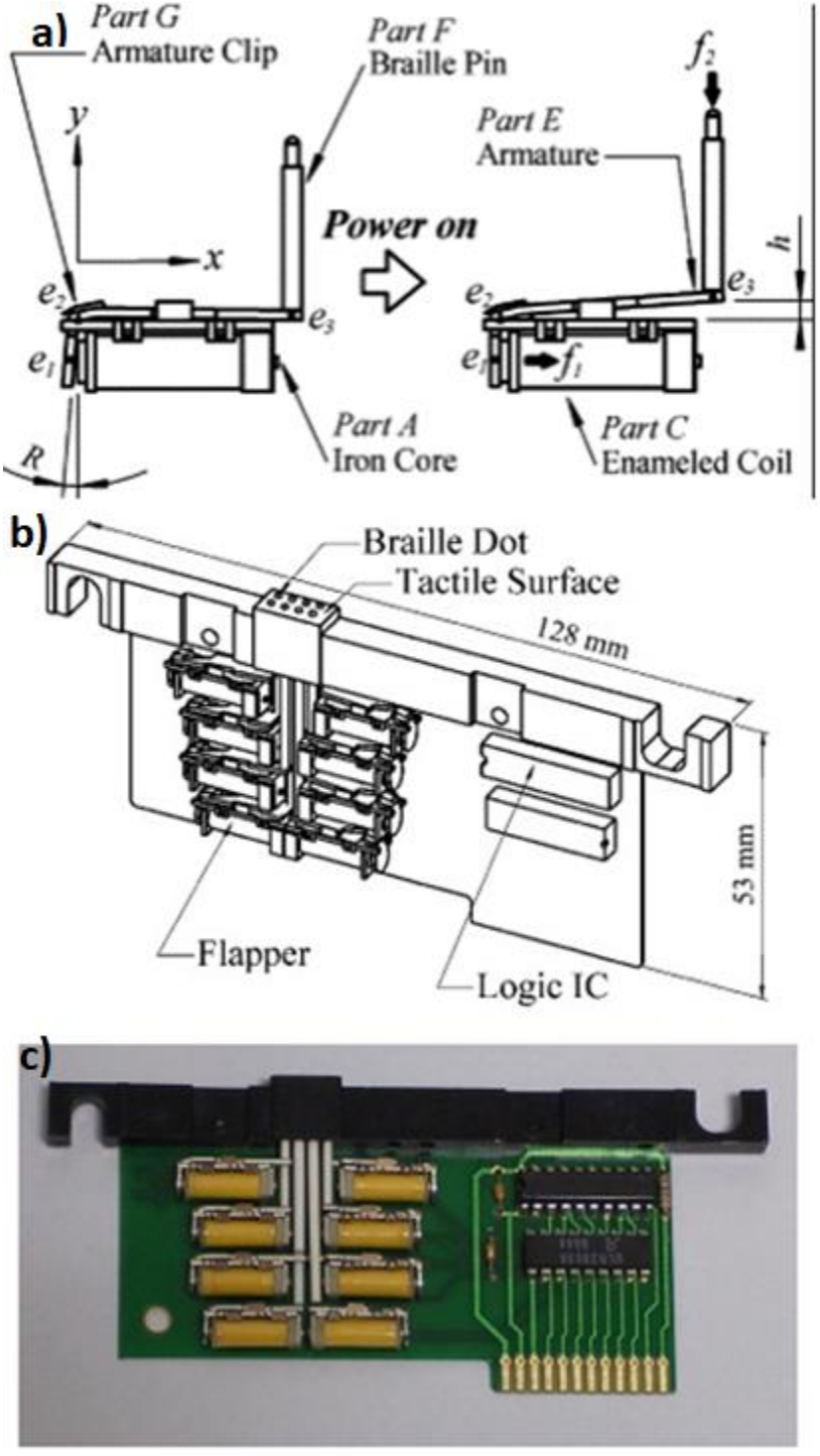


Figure 12. Combinational mechanism [18].

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:

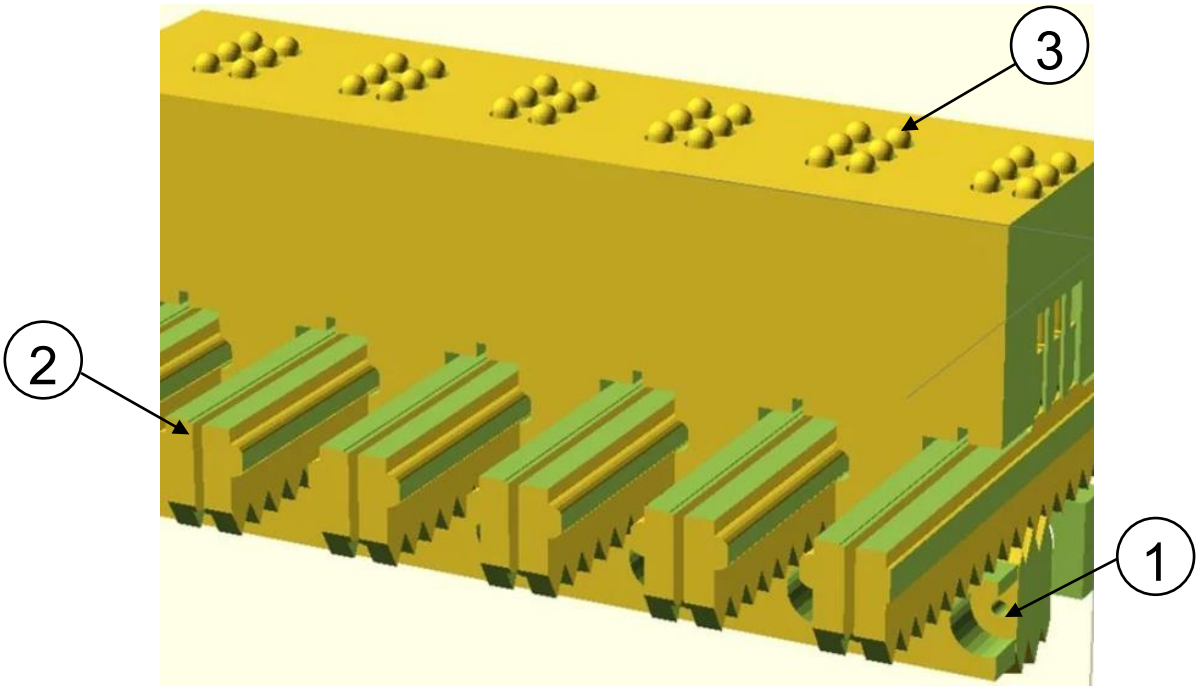
Mechanism is imprintable in ordinary FDM 3D printers, since parts are too tiny.

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).

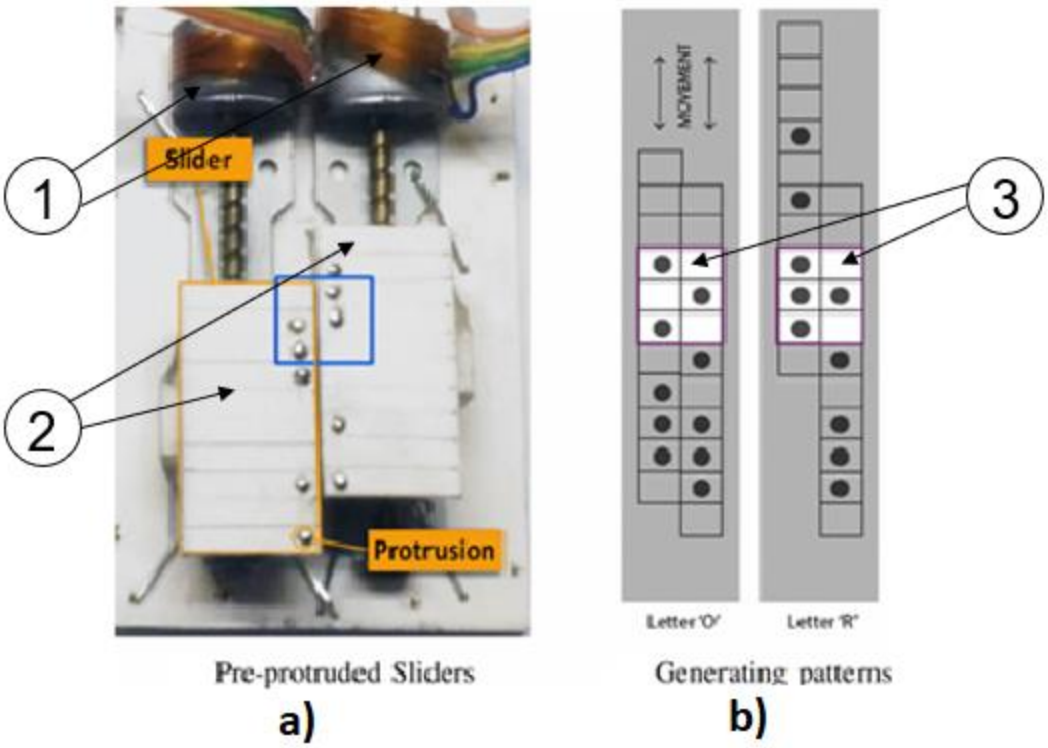


1. Circular motor place.
2. Sliding mechanism.
3. Pin.

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].

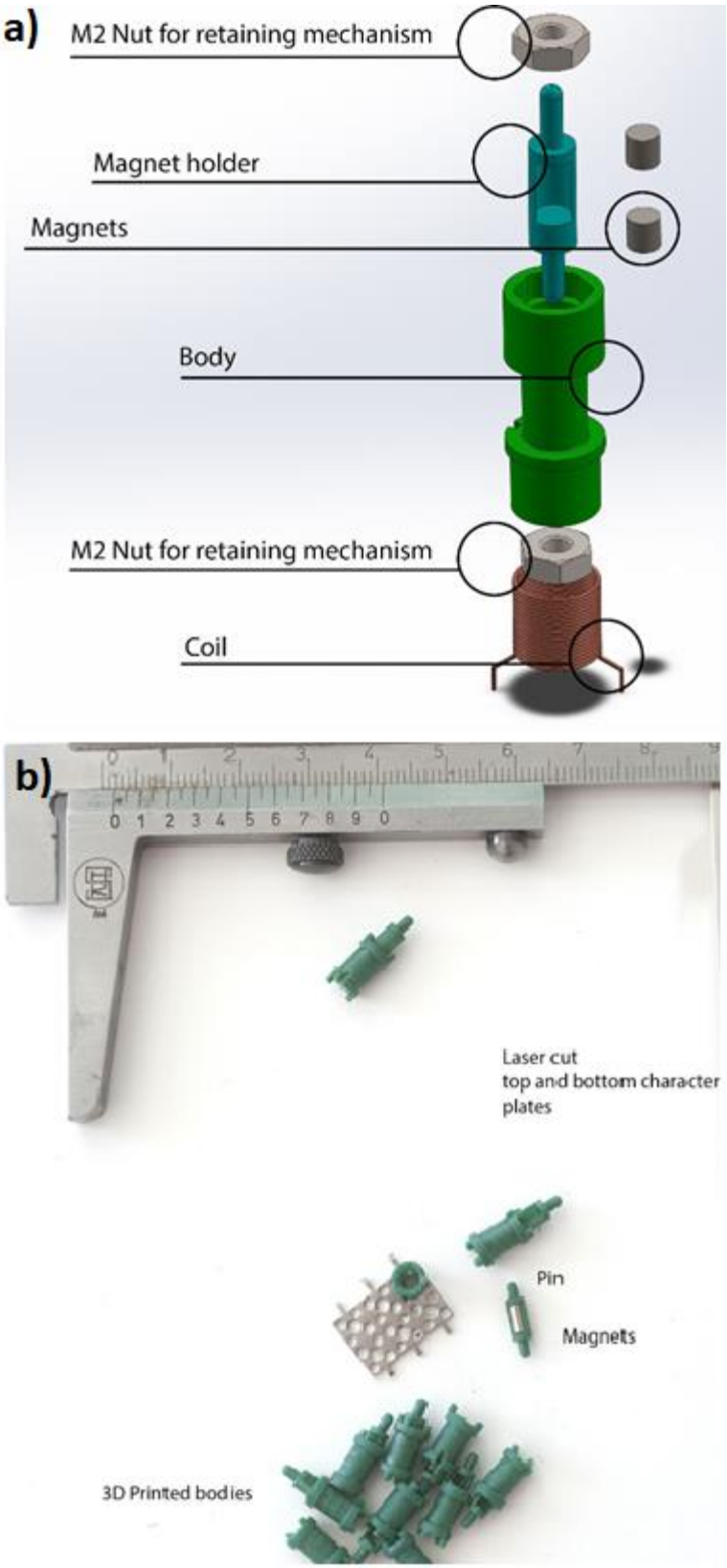


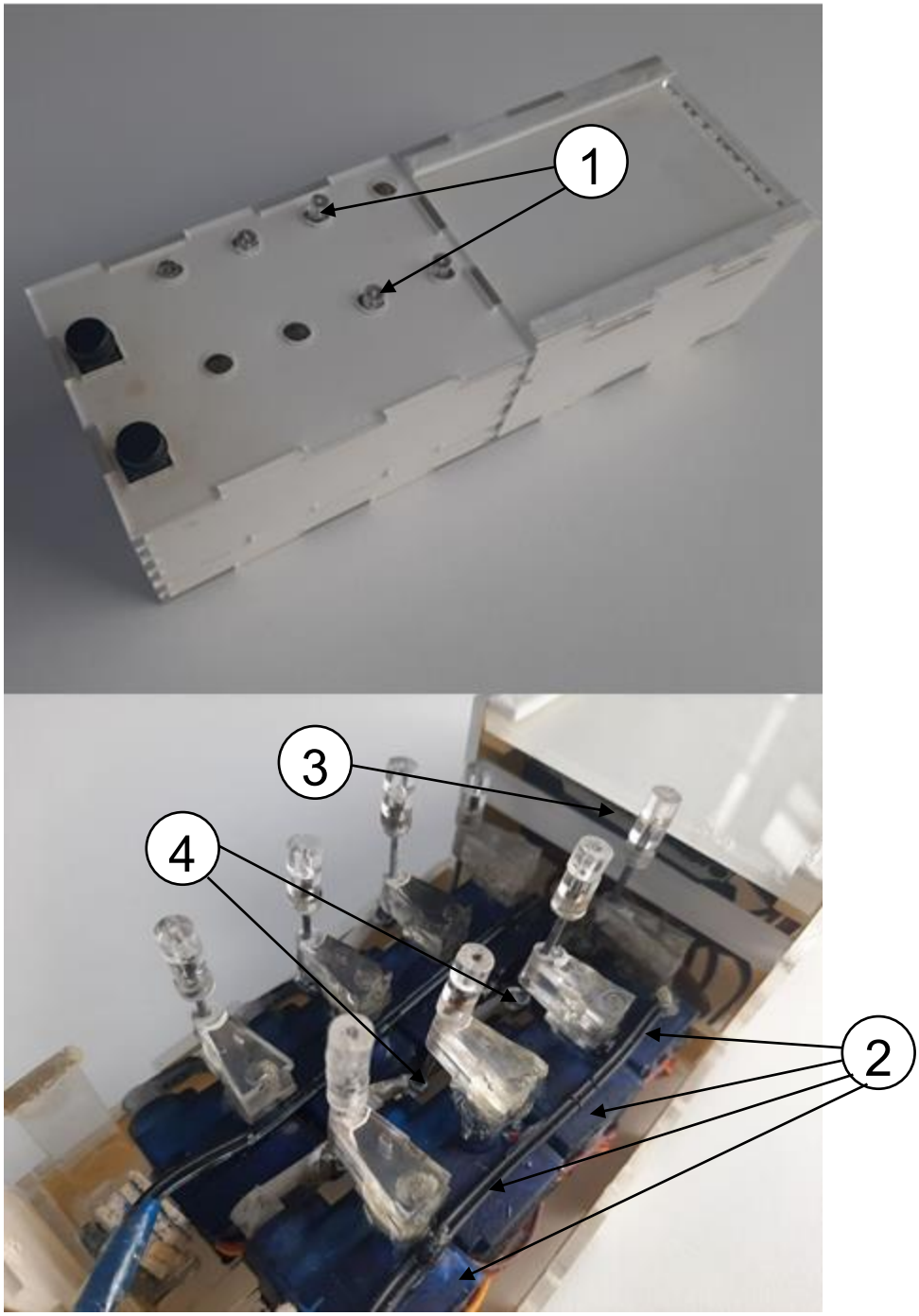
Figure 15. 3D parts of Modular Low-cost Braille Electronic Display [15].

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 Neodium 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)

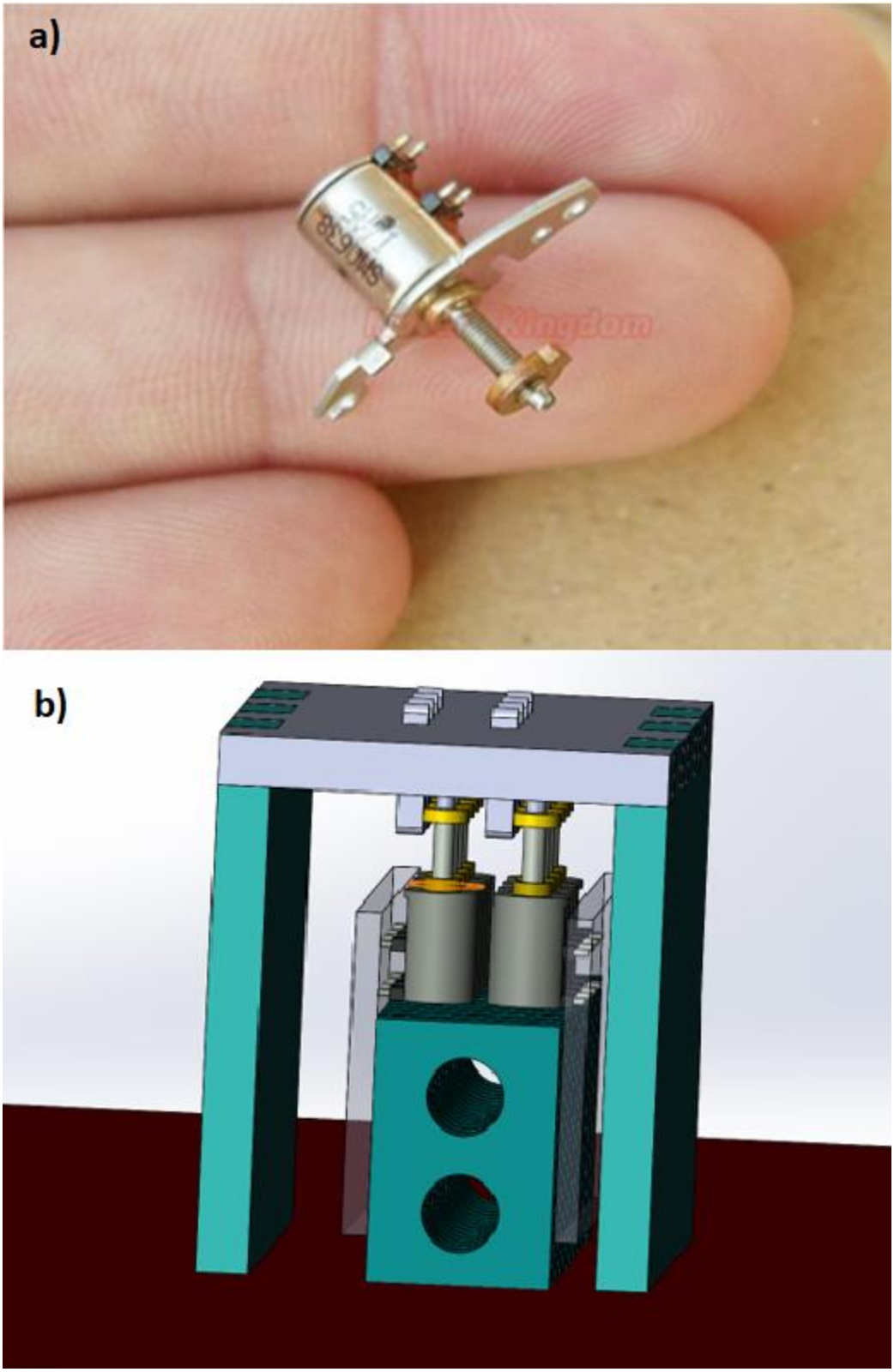


Figure 16. Braille display design with stepper motor.

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 one 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.

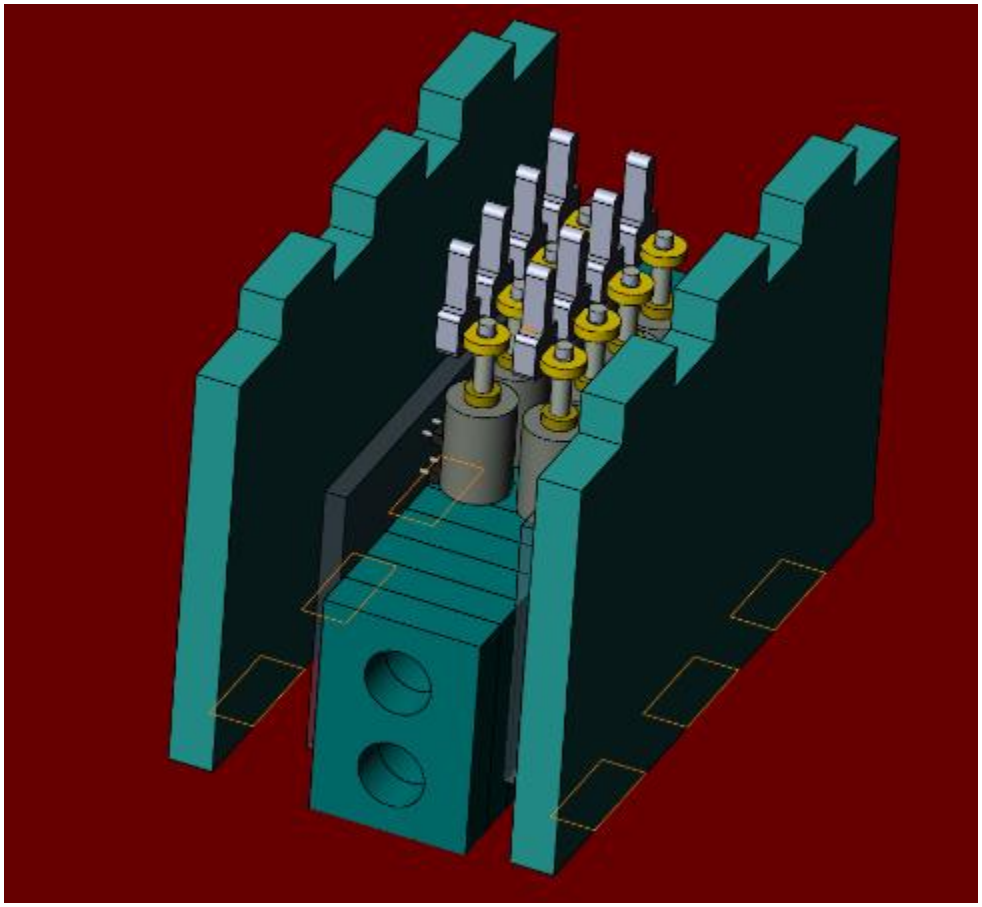


Figure 17. stepper motor view

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 fasting 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.

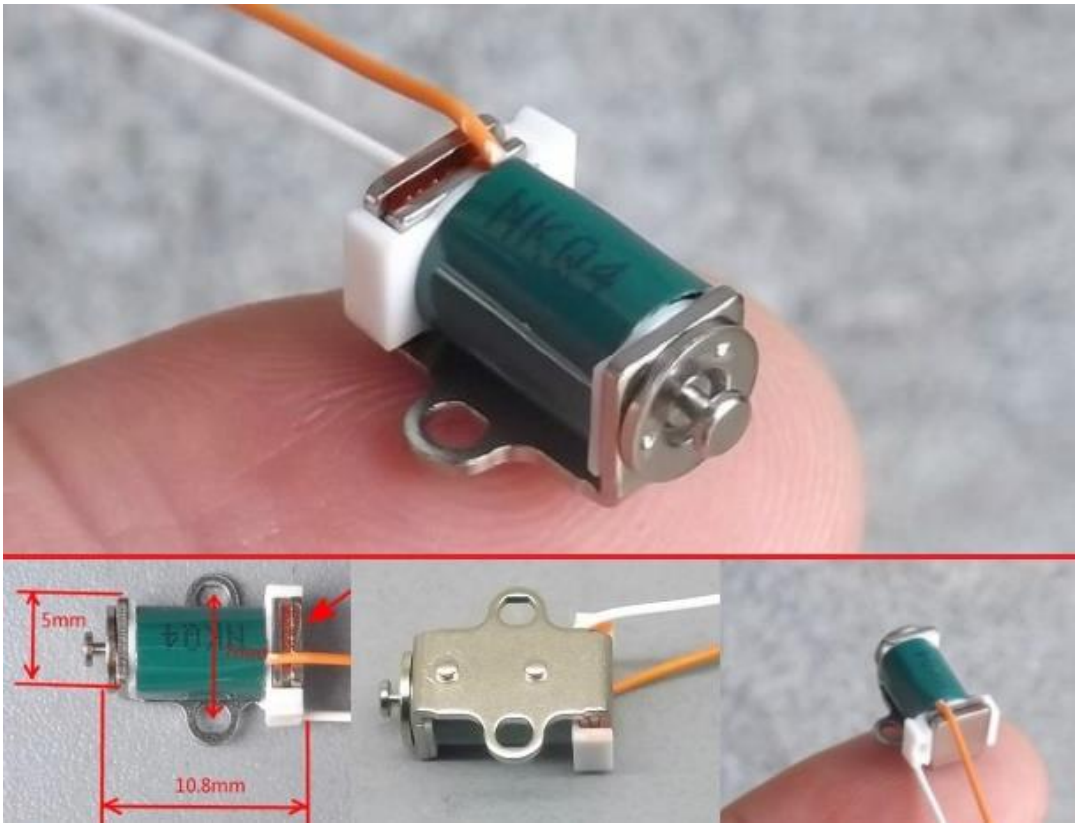


Figure 18. DC Electromagnetic Solenoid actuator [25].

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.

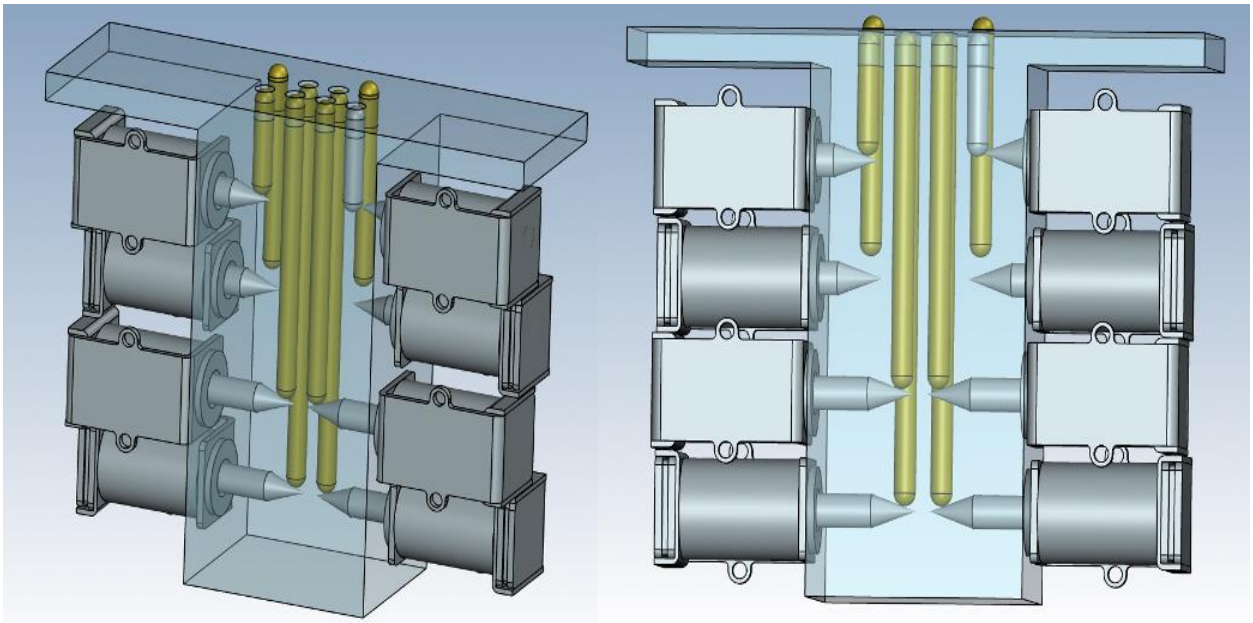


Figure 19. Conversion mechanism

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.

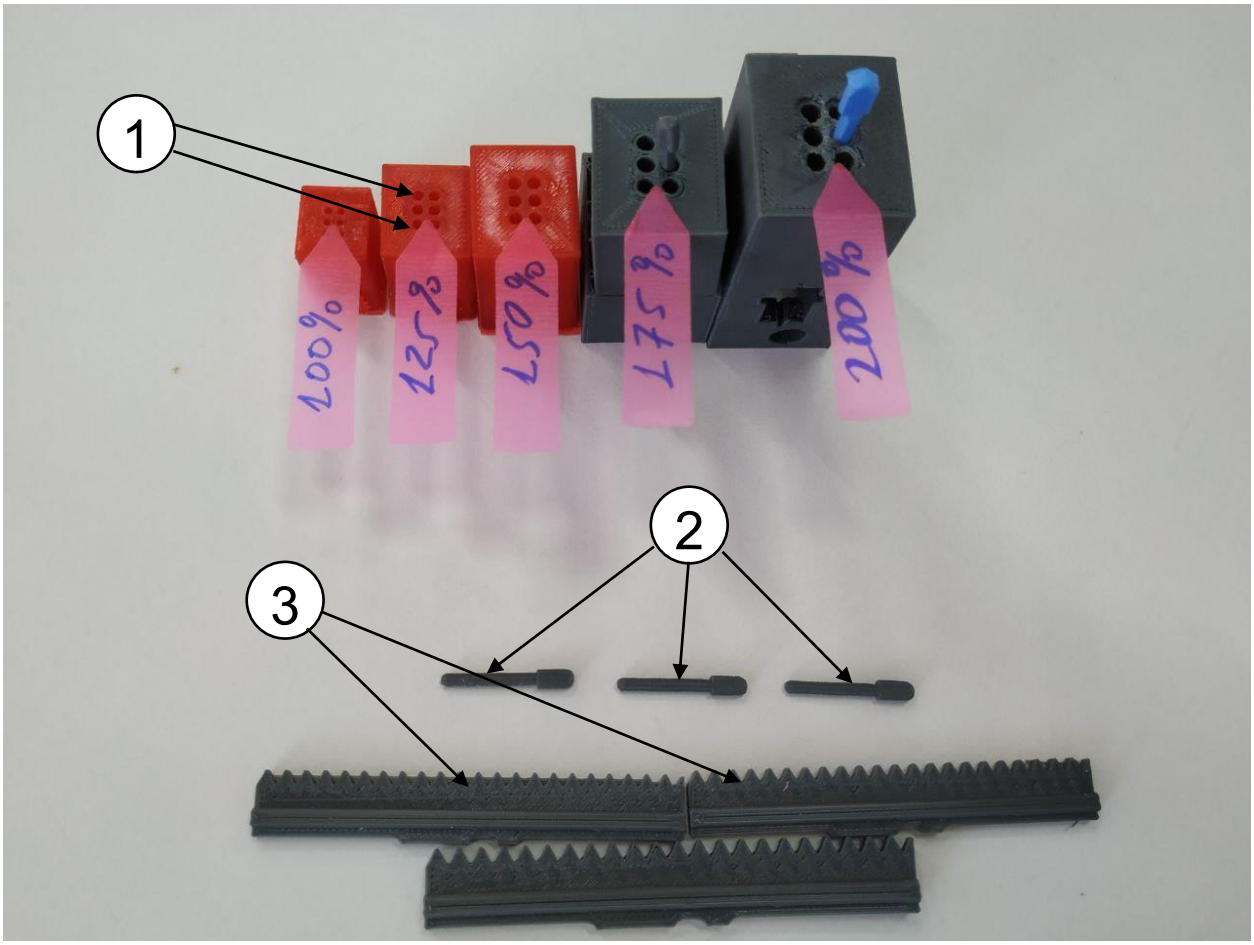
In the project various mechanisms were reverse engineered, to apply them in 3D printing, since there are no theoretical methods to analyze the FDM 3D prints, due to their complexity. Relying on laws of Machine Design and Structural Mechanics, followed with trial and error in practically printed 3D prints.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

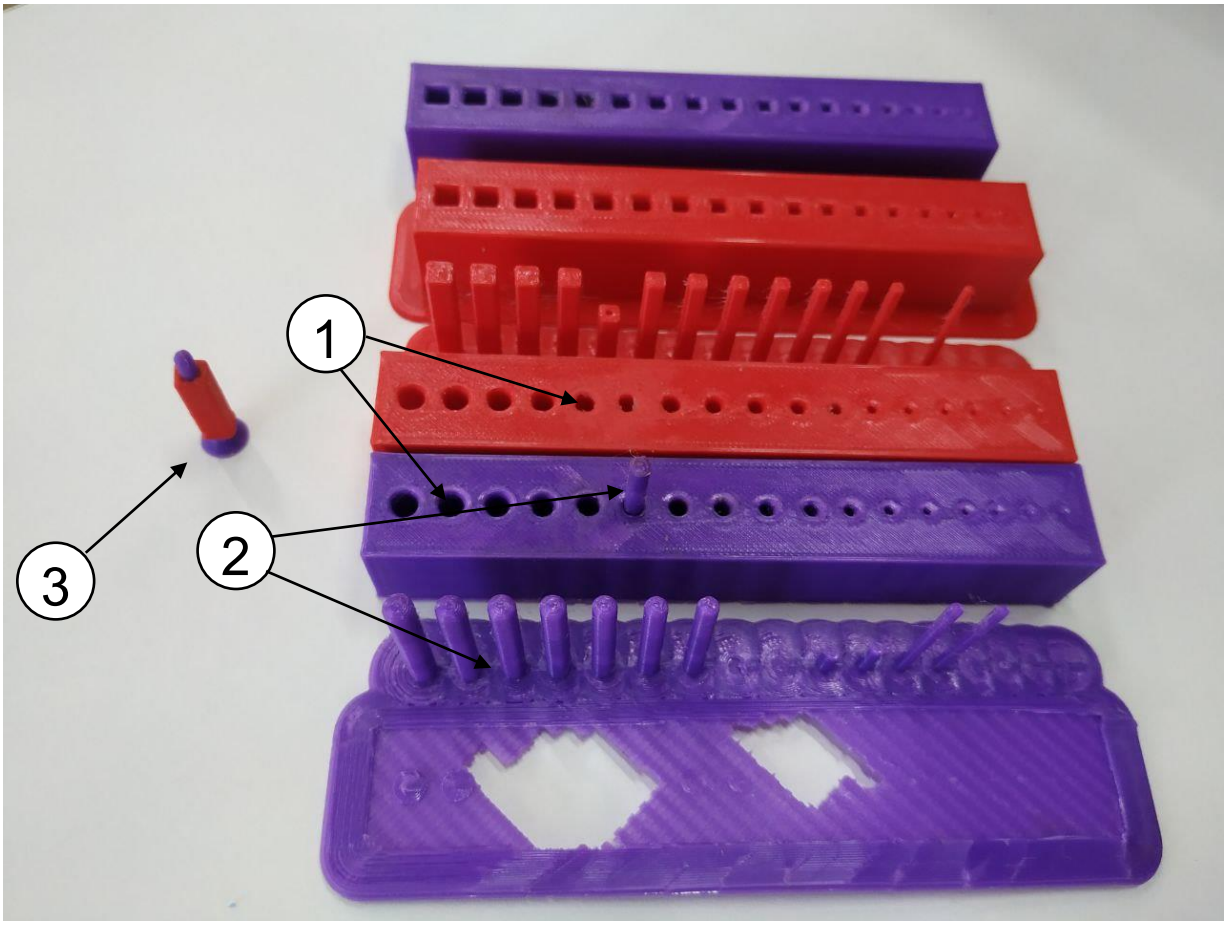


1. Holes.
2. Pins.
3. Racks.

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.

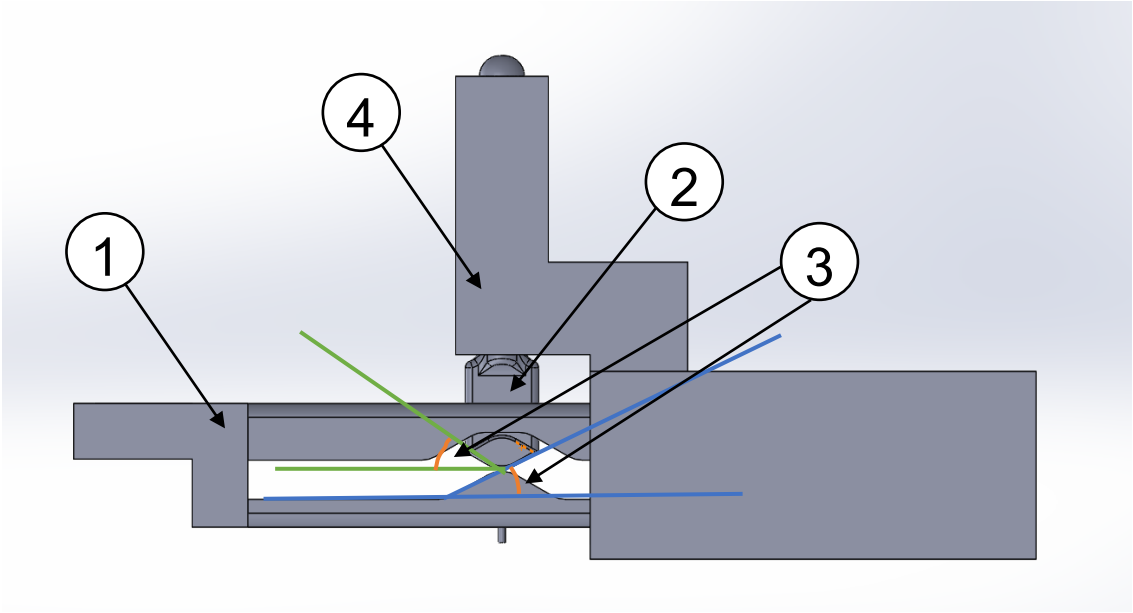


1 Holes.
2 Pins.
3 Chosen result of experience.

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

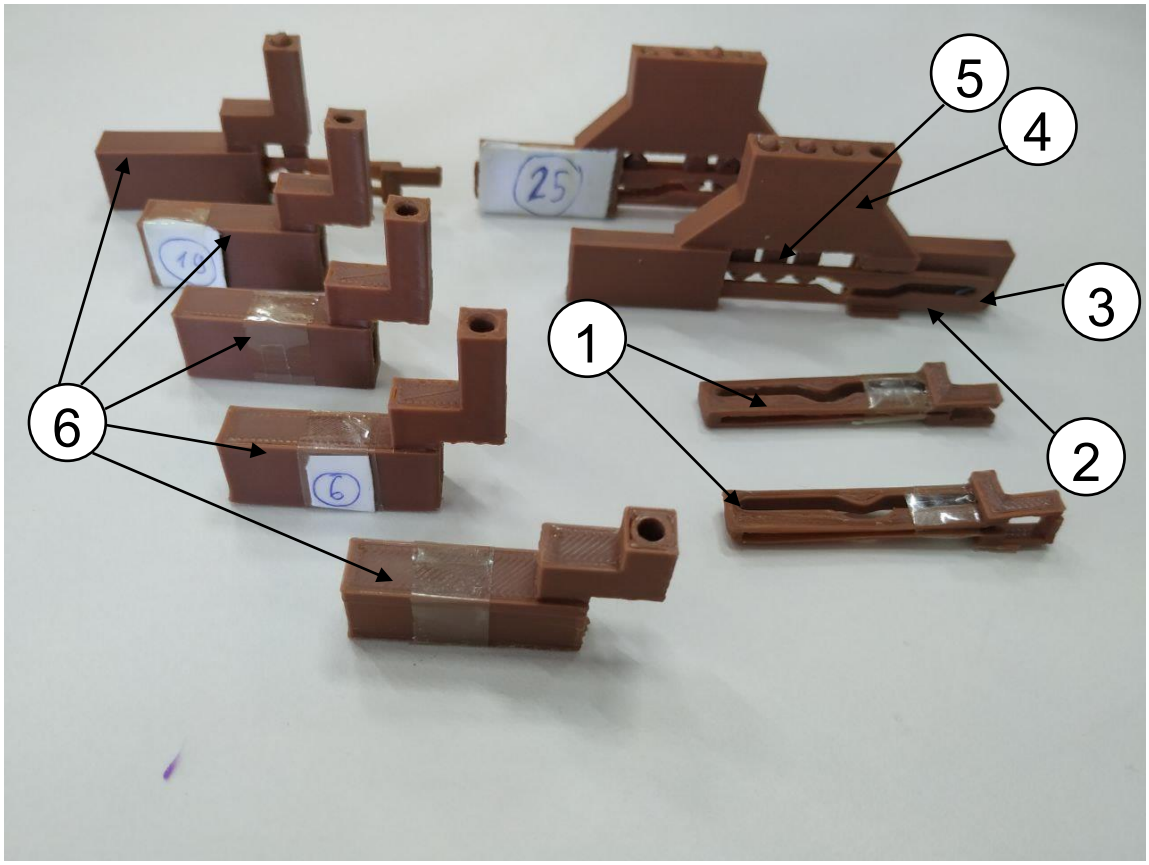
be the one that does not make one pin go out of its geometry (Figure 25). The case is, the ideal angle is those that tends to 0, but practically, the boundary is built by the pin geometry and 30 degrees is the best angle to keep the pin within the boundaries.



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.

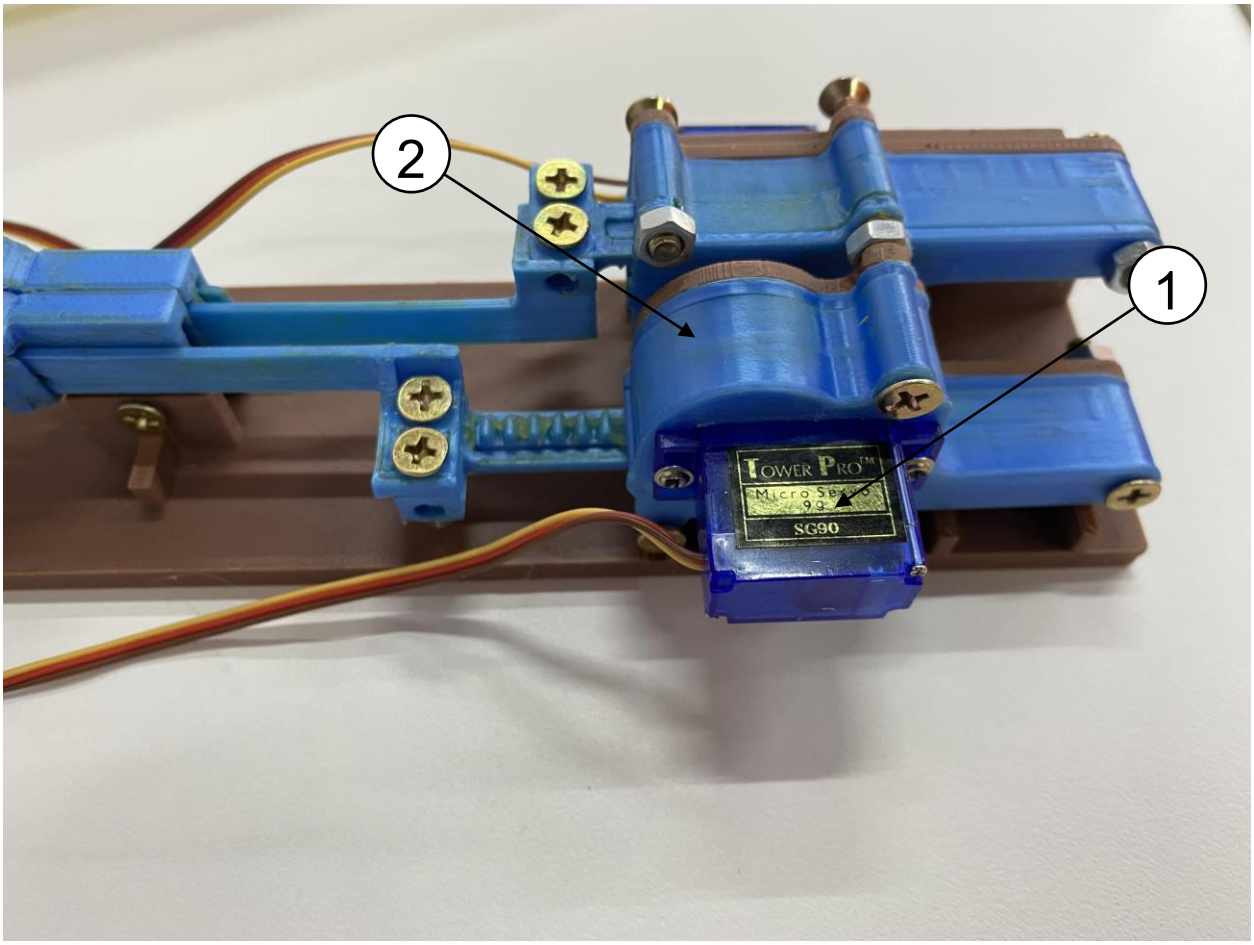


1. Rectangular parts (Sliding bars).
2. Horizontal part.
3. Vertical part.
4. Main block.
5. Rising pin.
6. Experiments to choose the right configuration between sliding bar and rising pin.

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.

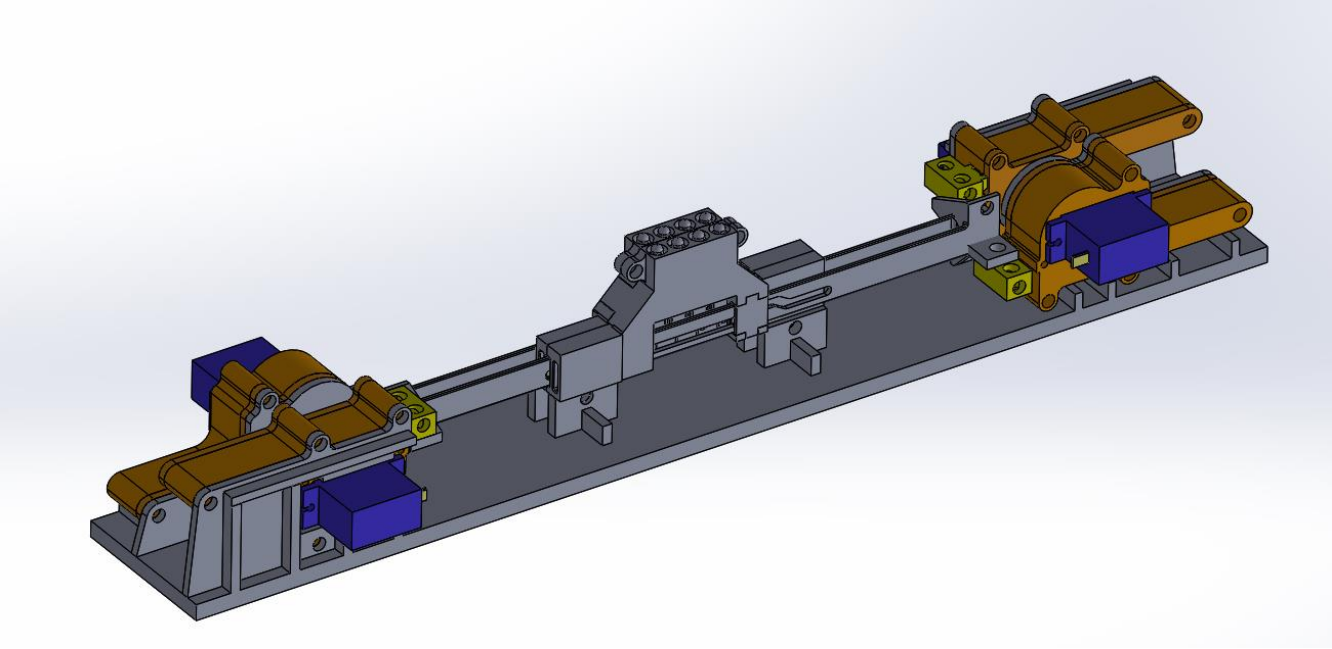
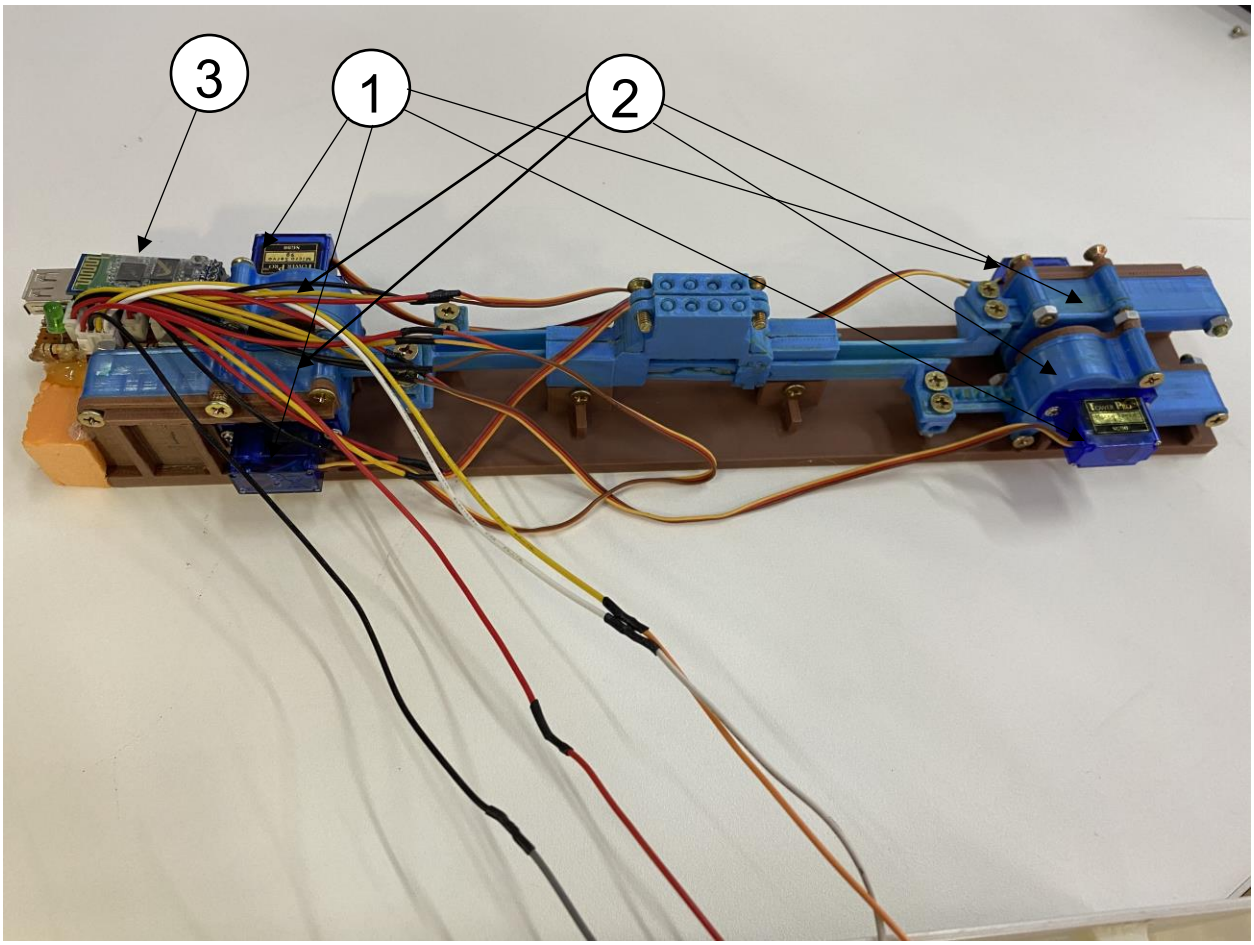


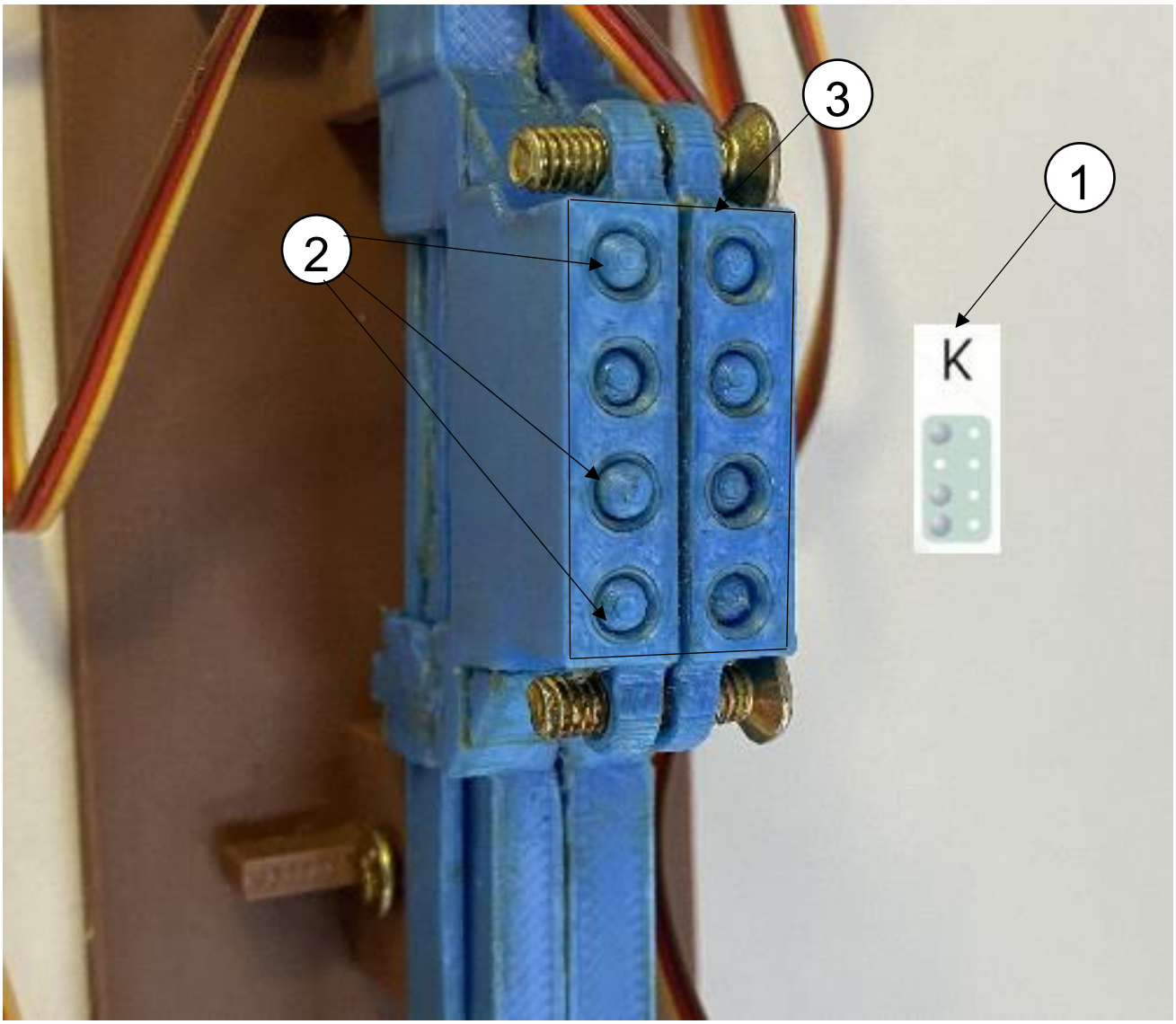
Figure 25. Final construction of 8 pins Braille display in 3d view.

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 Lenear actuators.
③ ATmega328P - 8-bit AVR Microcontroller. (Located under bluetooth module hc-05).

Figure 26. The overall view of the braille cell.



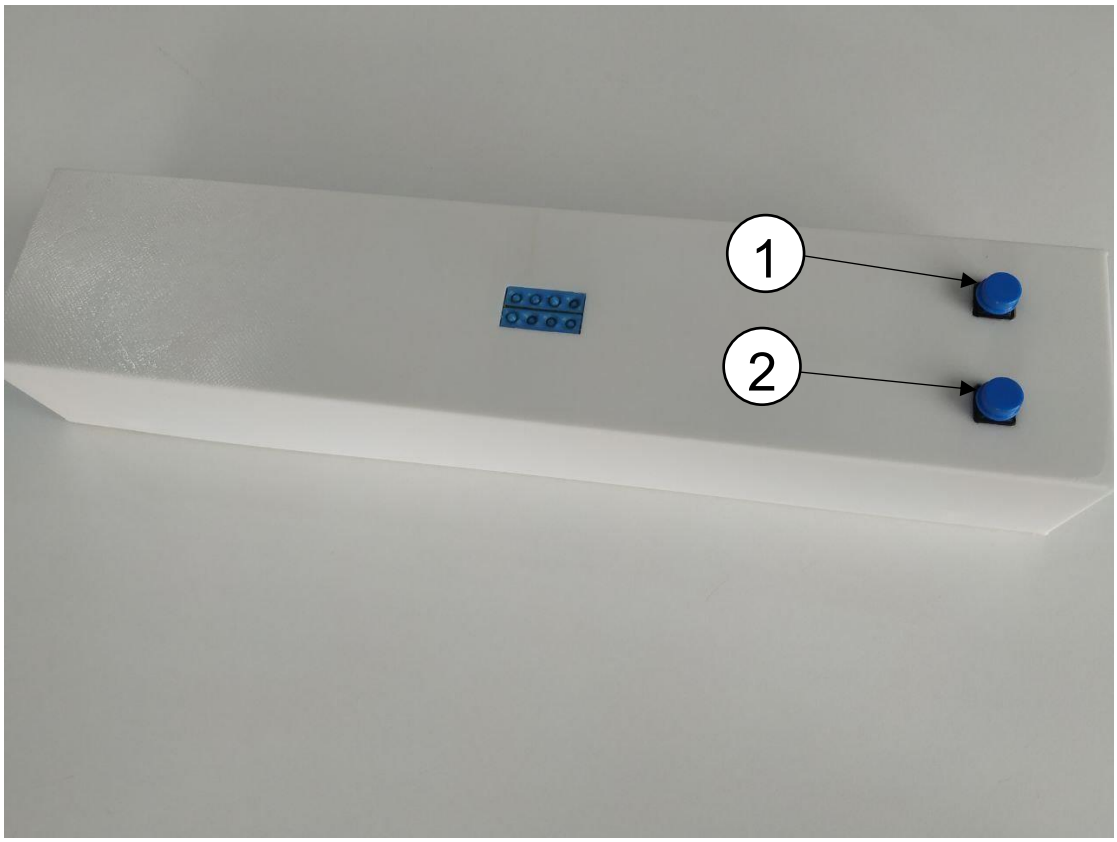
① ‘K’ in Braille Alpahabet.
② ‘K’ in Braille cells.
③ Braille cells.

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).

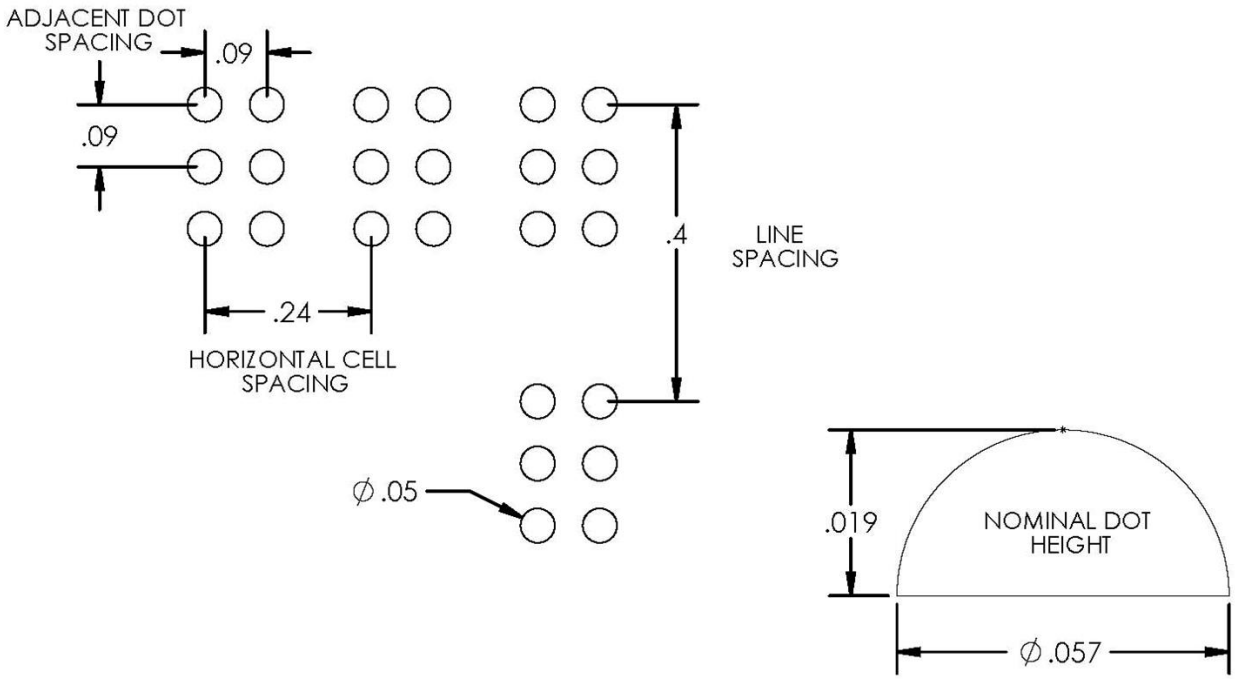


Figure 29. Braille size standards[26].

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=Cgdnd3Mtd2l6EAMYATICCAAyBwgAEIcCEBQyBggAEBYQHjoHCAAQRxCwAzoHCCMQ6gIQJzoECCMQJzoICC4QxwEQowI6AggAOggILhDHARCvATоFCAAQkQI6BAgAEEM6CAgAEMkDEJECUMbpBFiDywVgot8FaAFwA3gAgAGDAogB9xiSAQYwLjE1LjSYAQGgAQGqAQdnd3Mtd2l6sAEKyAEIwAEB&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://brailler.perkins.org/collections/perkins-braillers).
[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/hus/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. Ashraful 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] pyrophreek, “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/brailler-schematic1.png).

8.Appendix.

Appendix 1.

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

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

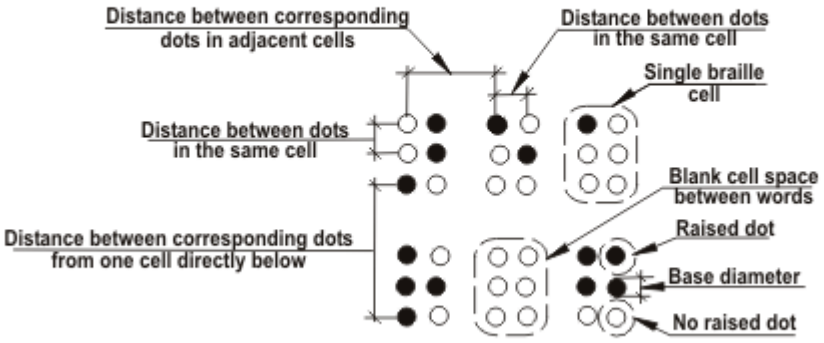


Figure 703.3.1
Braille Measurement

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

```
#include <Servo.h>
//Variables
Servo ser1;
Servo ser2;
Servo ser3;
Servo ser4;
Servo ser5;
Servo ser6;
Servo ser7;
Servo ser8;
int nextBut = 22;
int prevBut = 23;
String sampleText = "Hello"; /*Input texts*/
void setup() {
 // initializing variables
  ser1.attach(13);
  ser2.attach(12);
  ser3.attach(11);
  ser4.attach(10);
  ser5.attach(9);
  ser6.attach(8);
  ser7.attach(7);
  ser8.attach(6);
  pinMode(nextBut, INPUT);
  pinMode(prevBut, INPUT);
/*Access to 'reset' function and
 resetting all variables*/
  reset();
}
void loop() {
  //Initializing loop variables
  int i = 0;
  boolean done = true;
  //executing inputted text characters
  do {
/*Return to the initial position when
the first character was displayed in the text
only the previous button is pressed */
    if (!done && i == 0) {
      reset();
      done = true;
    }
/*Converting characters to braille system
and dispalying it */
    if (!done) {
      convert(sampleText[i - 1]);
      displayChar();
      done = true;
    }
/*Access to 'next' function
and alogirtm for loop variables*/
    if (next() == HIGH) {
      done = false;
      i++;
    }
/*Access to 'prev' function
and alogirtm for loop variables*/
    else if (prev() == HIGH) {
      done = false;
      i--;
    }
  } while (i <= sampleText.length());
 /*Access to reset function and
 resetting all variables*/
  reset();
}
// Debouncing buttons
boolean debaunce(boolean last, int pin) {
  int current = digitalRead(pin);
  if (last != current) {
    delay(5);
    current = digitalRead(pin);
  }
  return current;
}
//Initialized variables out of loop
boolean nextLast = LOW;
boolean prevLast = LOW;
int state[8];
//Controling next button
boolean next() {
  boolean current = debaunce(nextLast, nextBut);
  if (nextLast == LOW && current == HIGH) {
    nextLast = current;

    return HIGH;
  }
  nextLast = current;
  return LOW;
}
//Controling previous button
boolean prev() {
  boolean current = debaunce(prevLast, prevBut);
  if (prevLast == LOW && current == HIGH) {
    prevLast = current;
    return HIGH;
  }
  prevLast = current;
  return LOW;
}
//reset function
void reset() {
  ser2.write(115);
  ser5.write(118);
  ser6.write(115);
  ser1.write(110);
  ser3.write(112);
  ser4.write(112);
  ser7.write(117);
  ser8.write(120);
  state[0] = 0;
  state[1] = 0;
  state[2] = 0;
  state[3] = 0;
  state[4] = 0;
  state[5] = 0;
  state[6] = 0;
  state[7] = 0;
  delay(100);
}
/*Converting characters to
the braille system by this function*/
void convert(char c) {
  switch (c) {
    //Small letters
    case 'a': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 0;
        state[3] = 0;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'b': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 0;
        state[3] = 0;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }

    case 'c': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 0;
        state[3] = 1;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'd': {
```

```
        state[0] = 1;
        state[1] = 0;
        state[2] = 0;
        state[3] = 1;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'e': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 0;
        state[3] = 0;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'f': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 0;
        state[3] = 1;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'g': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 0;
        state[3] = 1;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'h': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 0;
        state[3] = 0;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'i': {
        state[0] = 0;
        state[1] = 1;
        state[2] = 0;
        state[3] = 1;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'j': {
        state[0] = 0;
        state[1] = 1;
        state[2] = 0;
        state[3] = 1;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'k': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 1;
        state[3] = 0;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'l': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 1;
        state[3] = 0;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'm': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 1;
        state[3] = 1;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'n': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 1;
        state[3] = 1;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'o': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 1;
        state[3] = 0;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'p': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 1;
        state[3] = 1;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'q': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 1;
        state[3] = 1;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'r': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 1;
        state[3] = 0;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 's': {
        state[0] = 0;
        state[1] = 1;
        state[2] = 1;
        state[3] = 1;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 't': {
        state[0] = 0;
        state[1] = 1;
        state[2] = 1;
        state[3] = 1;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'u': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 1;
        state[3] = 0;
        state[4] = 0;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'v': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 1;
        state[3] = 0;
        state[4] = 0;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'w': {
        state[0] = 0;
        state[1] = 1;
        state[2] = 0;
        state[3] = 1;
        state[4] = 1;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'x': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 1;
        state[3] = 1;
        state[4] = 0;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case 'y': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 1;
        state[3] = 1;
        state[4] = 1;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
```

```
    }
  case 'D': {
      state[0] = 1;
      state[1] = 0;
      state[2] = 0;
      state[3] = 1;
      state[4] = 1;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'E': {
      state[0] = 1;
      state[1] = 0;
      state[2] = 0;
      state[3] = 0;
      state[4] = 1;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'F': {
      state[0] = 1;
      state[1] = 1;
      state[2] = 0;
      state[3] = 1;
      state[4] = 0;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'G': {
      state[0] = 1;
      state[1] = 1;
      state[2] = 0;
      state[3] = 1;
      state[4] = 1;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'H': {
      state[0] = 1;
      state[1] = 1;
      state[2] = 0;
      state[3] = 0;
      state[4] = 1;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'I': {
      state[0] = 0;
      state[1] = 1;
      state[2] = 0;
      state[3] = 1;
      state[4] = 0;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'J': {
      state[0] = 0;
      state[1] = 1;
      state[2] = 0;
      state[3] = 1;
      state[4] = 1;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'K': {
      state[0] = 1;
      state[1] = 0;
      state[2] = 1;
      state[3] = 0;
      state[4] = 0;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'L': {
      state[0] = 1;
      state[1] = 1;
      state[2] = 1;
      state[3] = 0;
      state[4] = 0;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'M': {
      state[0] = 1;
      state[1] = 0;
      state[2] = 1;
      state[3] = 1;
      state[4] = 0;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'N': {
      state[0] = 1;
      state[1] = 0;
      state[2] = 1;
      state[3] = 1;
      state[4] = 1;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'O': {
      state[0] = 1;
      state[1] = 0;
      state[2] = 1;
      state[3] = 0;
      state[4] = 1;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'P': {
      state[0] = 1;
      state[1] = 1;
      state[2] = 1;
      state[3] = 1;
      state[4] = 0;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'Q': {
      state[0] = 1;
      state[1] = 1;
      state[2] = 1;
      state[3] = 1;
      state[4] = 1;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'R': {
      state[0] = 1;
      state[1] = 1;
      state[2] = 1;
      state[3] = 0;
      state[4] = 1;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'S': {
      state[0] = 0;
      state[1] = 1;
      state[2] = 1;
      state[3] = 1;
      state[4] = 0;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'T': {
      state[0] = 0;
      state[1] = 1;
      state[2] = 1;
      state[3] = 1;
      state[4] = 1;
      state[5] = 0;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'U': {
      state[0] = 1;
      state[1] = 0;
      state[2] = 1;
      state[3] = 0;
      state[4] = 0;
      state[5] = 1;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'V': {
      state[0] = 1;
      state[1] = 1;
      state[2] = 1;
      state[3] = 0;
      state[4] = 0;
      state[5] = 1;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'W': {
      state[0] = 0;
      state[1] = 1;
      state[2] = 0;
      state[3] = 1;
      state[4] = 1;
      state[5] = 1;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'X': {
      state[0] = 1;
      state[1] = 0;
      state[2] = 1;
      state[3] = 1;
      state[4] = 0;
      state[5] = 1;
      state[6] = 1;
      state[7] = 0;
      break;
    }
  case 'Y': {
      state[0] = 1;
      state[1] = 0;
      state[2] = 1;
      state[3] = 1;
      state[4] = 1;
      state[5] = 1;
      state[6] = 1;
```

```
        state[7] = 0;
        break;
      }
    case 'Z': {
        state[0] = 1;
        state[1] = 0;
        state[2] = 1;
        state[3] = 0;
        state[4] = 1;
        state[5] = 1;
        state[6] = 1;
        state[7] = 0;
        break;
      }
    case 'Ä': {
        state[0] = 0;
        state[1] = 0;
        state[2] = 1;
        state[3] = 1;
        state[4] = 1;
        state[5] = 0;
        state[6] = 1;
        state[7] = 1;
        break;
      }
    case 'Ë': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 0;
        state[3] = 1;
        state[4] = 0;
        state[5] = 1;
        state[6] = 1;
        state[7] = 1;
        break;
      }
    case 'É': {
        state[0] = 1;
        state[1] = 1;
        state[2] = 1;
        state[3] = 1;
        state[4] = 1;
        state[5] = 1;
        state[6] = 1;
        state[7] = 1;
        break;
      }

    // Numbers
    case '0':  {
        state[0] = 0;
        state[1] = 0;
        state[2] = 1;
        state[3] = 1;
        state[4] = 1;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '1':  {
        state[0] = 1;
        state[1] = 0;
        state[2] = 0;
        state[3] = 0;
        state[4] = 0;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '2':  {
        state[0] = 1;
        state[1] = 1;
        state[2] = 0;
        state[3] = 0;
        state[4] = 0;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '3':  {
        state[0] = 1;
        state[1] = 0;
        state[2] = 0;
        state[3] = 1;
        state[4] = 0;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '4':  {
        state[0] = 1;
        state[1] = 0;
        state[2] = 0;
        state[3] = 1;
        state[4] = 1;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '5':  {
        state[0] = 1;
        state[1] = 0;
        state[2] = 0;
        state[3] = 0;
        state[4] = 1;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '6':  {
        state[0] = 1;
        state[1] = 1;
        state[2] = 0;
        state[3] = 1;
        state[4] = 0;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '7':  {
        state[0] = 1;
        state[1] = 1;
        state[2] = 0;
        state[3] = 1;
        state[4] = 1;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '8':  {
        state[0] = 1;
        state[1] = 1;
        state[2] = 0;
        state[3] = 0;
        state[4] = 1;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '9':  {
        state[0] = 0;
        state[1] = 1;
        state[2] = 0;
        state[3] = 1;
        state[4] = 0;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }

    //  Punctuation
    case ' ':  {
        state[0] = 0;
        state[1] = 0;
        state[2] = 0;
        state[3] = 0;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case ',':  {
        state[0] = 0;
        state[1] = 1;
        state[2] = 0;
        state[3] = 0;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '.':  {
        state[0] = 0;
        state[1] = 0;
        state[2] = 1;
        state[3] = 0;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '!':  {
        state[0] = 0;
        state[1] = 0;
        state[2] = 0;
        state[3] = 0;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '?':  {
        state[0] = 0;
        state[1] = 1;
        state[2] = 0;
        state[3] = 0;
        state[4] = 0;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case ';':  {
        state[0] = 0;
        state[1] = 1;
        state[2] = 1;
        state[3] = 0;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case ':':  {
        state[0] = 0;
        state[1] = 1;
        state[2] = 0;
        state[3] = 0;
        state[4] = 1;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '"':  {
        state[0] = 0;
```

```
        state[1] = 0;
        state[2] = 0;
        state[3] = 1;
        state[4] = 0;
        state[5] = 0;
        state[6] = 0;
        state[7] = 0;
        break;
      }

    case '(':  {
        state[0] = 0;
        state[1] = 1;
        state[2] = 1;
        state[3] = 0;
        state[4] = 0;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case ')':  {
        state[0] = 0;
        state[1] = 0;
        state[2] = 1;
        state[3] = 0;
        state[4] = 1;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '{':  {
        state[0] = 1;
        state[1] = 1;
        state[2] = 1;
        state[3] = 0;
        state[4] = 1;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '}':  {
        state[0] = 0;
        state[1] = 1;
        state[2] = 1;
        state[3] = 1;
        state[4] = 1;
        state[5] = 1;
        state[6] = 0;
        state[7] = 0;
        break;
      }
    case '[':  {
        state[0] = 1;
        state[1] = 1;
        state[2] = 1;
        state[3] = 0;
        state[4] = 1;
        state[5] = 1;
        state[6] = 1;
        state[7] = 0;
        break;
      }
    case ']':  {
        state[0] = 0;
        state[1] = 1;
        state[2] = 1;
        state[3] = 1;
        state[4] = 1;
        state[5] = 1;
        state[6] = 1;
        state[7] = 0;
        break;
      }
    case '/':  {
        state[0] = 1;
        state[1] = 1;
        state[2] = 1;
        state[3] = 1;
        state[4] = 1;
        state[5] = 1;
        state[6] = 1;
        state[7] = 1;
        break;
       }
  }
}
/*Displaying character which is
converted to braille system*/
void displayChar() {
  if (state[0])    //1st servo
  {
    ser1.write(65);
  }
  else
  {
    ser1.write(115);
  }
  if (state[1])     //2nd servo
  {
    ser2.write(75);
  }
  else
  {
    ser2.write(115);
  }
  if (state[2])     //3rd servo
  {
    ser3.write(65);
  }
  else
  {
    ser3.write(112);
  }
  if (state[3])      //4th servo
  {
    ser4.write(66);
  }
  else
  {
    ser4.write(112);
  }
  if (state[4])      //5th servo
  {
    ser5.write(70);
  }
  else
  {
    ser5.write(118);
  }
  if (state[5])      //6th servo
  {
    ser6.write(70);
  }
  else
  {
    ser6.write(110);
  }
  if (state[6])      //7th servo
  {
    ser7.write(65);
  }
  else
  {
    ser7.write(117);
  }
  if (state[7])      //8th servo
  {
    ser8.write(60);
  }
  else
  {
    ser8.write(120);
  }
  delay(100);
}
```

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.

```
#include <Servo.h>
//Variables
Servo ser1;
Servo ser2;
Servo ser3;
Servo ser4;
int nextBut = 2;
int prevBut = 4;
String sampleText;
void setup() {
  // initializing variables
  Serial.begin(9600);
  ser1.attach(6);
  ser2.attach(9);
  ser3.attach(10);
  ser4.attach(11);
  pinMode(nextBut, INPUT);
  pinMode(prevBut, INPUT);
  /*Access to 'reset' function and
    resetting all variables*/
  reset();
}
void loop() {
  //Initializing loop variables
  int i = 0;
  boolean done = true;
  /*To allows input from console to program
    via "Serial monitor*/
  while (Serial.available() == 0)
  {
    reset();
  }
  /*read string from serial monitor*/
  String sampleText = Serial.readString();
  //executing inputted text characters
  do {
    /*After reading string from serial monitor
      it resets motors to initial position via stopping loop*/
    if (Serial.available() > 0)
    {
      break;
    }
    /*Return to the initial position when
      the first character was displayed in the text
      only the previous button is pressed */
    if (!done && i == 0) {
      reset();
      done = true;
    }
    /*Converting characters to braille system
      and dispalying it */
    if (!done) {
      convert(sampleText[i - 1]);
      displayChar();
      done = true;
    }
    /*Access to 'next' function
      and alogirtm for loop variables*/
    if (next() == HIGH) {
      done = false;
      i++;
    }
    /*Access to 'prev' function
      and alogirtm for loop variables*/
    else if (prev() == HIGH) {
      done = false;
      i--;
    }
  } while (i <= sampleText.length());
  /*Access to reset function and
    resetting all variables*/
  reset();
}
// Debouncing buttons
boolean debaunce(boolean last, int pin) {
  int current = digitalRead(pin);
  if (last != current) {
    delay(5);
    current = digitalRead(pin);
  }
  return current;
}
//Initialized variables out of loop
boolean nextLast = LOW;
boolean prevLast = LOW;
int state[12];
//Controling next button
boolean next() {
  boolean current = debaunce(nextLast, nextBut);
  if (nextLast == LOW && current == HIGH) {
    nextLast = current;

    return HIGH;
  }
  nextLast = current;
  return LOW;
}
//Controling previous button
boolean prev() {
  boolean current = debaunce(prevLast, prevBut);
  if (prevLast == LOW && current == HIGH) {
    prevLast = current;
    return HIGH;
  }
  prevLast = current;
  return LOW;
}
//reset function
void reset() {
  ser1.write(20);
  ser2.write(170);
  ser3.write(130);
  ser4.write(0);

  state[0] = 0;
  state[1] = 0;
  state[2] = 0;
  state[3] = 0;
  state[4] = 0;
  state[5] = 0;
  state[6] = 0;
  state[7] = 0;
  state[8] = 0;
  state[9] = 0;
  state[10] = 0;
  state[11] = 0;
  delay(100);
}
/*Converting characters to
  the braille system by this function*/
void convert(char c) {
  switch (c) {
    //Small letters
    case 'a': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'b': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

```

```
        state[6] = 0;  //3rd servo
        state[7] = 0;
        state[8] = 0;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'c': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'd': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0;

        state[9] = 0;  //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'e': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'f': {

        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'g': {

        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'h': {

        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;

        state[9] = 0;  //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'i': {

        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'j': {

        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'k': {

```

```
      state[0] = 1; //1st servo
      state[1] = 0;
      state[2] = 0;

      state[3] = 0; //2nd servo
      state[4] = 0;
      state[5] = 1;

      state[6] = 0; //3rd servo
      state[7] = 0;
      state[8] = 0 ;

      state[9] = 0; //4th servo
      state[10] = 0;
      state[11] = 0;
      break;
    }
  case 'l': {

      state[0] = 0; //1st servo
      state[1] = 1;
      state[2] = 0;

      state[3] = 0; //2nd servo
      state[4] = 0;
      state[5] = 1;

      state[6] = 0; //3rd servo
      state[7] = 0;
      state[8] = 0 ;

      state[9] = 0; //4th servo
      state[10] = 0;
      state[11] = 0;
      break;
    }
  case 'm': {

      state[0] = 1; //1st servo
      state[1] = 0;
      state[2] = 0;

      state[3] = 0; //2nd servo
      state[4] = 0;
      state[5] = 1;

      state[6] = 1; //3rd servo
      state[7] = 0;
      state[8] = 1 ;

      state[9] = 0; //4th servo
      state[10] = 0;
      state[11] = 0;
      break;
    }
  case 'n': {

      state[0] = 1; //1st servo
      state[1] = 0;
      state[2] = 0;

      state[3] = 0; //2nd servo
      state[4] = 0;
      state[5] = 1;

      state[6] = 0; //3rd servo
      state[7] = 1;
      state[8] = 1 ;

      state[9] = 0;  //4th servo
      state[10] = 0;
      state[11] = 0;
      break;
    }
  case 'o': {

      state[0] = 1; //1st servo
      state[1] = 0;
      state[2] = 0;

      state[3] = 0; //2nd servo
      state[4] = 0;
      state[5] = 1;

      state[6] = 0; //3rd servo
      state[7] = 0;
      state[8] = 1 ;

      state[9] = 0;  //4th servo
      state[10] = 0;
      state[11] = 0;
      break;
    }
  case 'p': {

      state[0] = 0;  //1st servo
      state[1] = 1;
      state[2] = 0;

      state[3] = 0;  //2nd servo
      state[4] = 0;
      state[5] = 1;

      state[6] = 1; //3rd servo
      state[7] = 0;
      state[8] = 0 ;

      state[9] = 0; //4th servo
      state[10] = 0;
      state[11] = 0;
      break;
    }
  case 'q': {

      state[0] = 0; //1st servo
      state[1] = 1;
      state[2] = 0;

      state[3] = 0;  //2nd servo
      state[4] = 0;
      state[5] = 1;

      state[6] = 0;  //3rd servo
      state[7] = 1;
      state[8] = 0 ;

      state[9] = 0; //4th servo
      state[10] = 0;
      state[11] = 0;
      break;
    }
  case 'r': {

      state[0] = 0; //1st servo
      state[1] = 1;
      state[2] = 0;

      state[3] = 0; //2nd servo
      state[4] = 0;
      state[5] = 1;

      state[6] = 0; //3rd servo
      state[7] = 0;
      state[8] = 1 ;

      state[9] = 0;  //4th servo
      state[10] = 0;
      state[11] = 0;
      break;
    }
  case 's': {

      state[0] = 0; //1st servo
      state[1] = 0;
      state[2] = 1;

      state[3] = 0; //2nd servo
      state[4] = 0;
      state[5] = 1;
```

```
        state[6] = 1;  //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0;  //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 't': {
        state[0] = 0;  //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'u': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case 'v': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case 'w': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0;  //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 3;

      }
    case 'x': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0;  //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case 'y': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case 'z': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    //  CAPITAL LATTERS

    case 'A': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 1; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0;

        state[9] = 0;  //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'B': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;
```

```
        state[3] = 1; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'C': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 1; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'D': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 1;  //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0;  //3rd servo
        state[7] = 1;
        state[8] = 0;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'E': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 1; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'F': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 1; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'G': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 1; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'H': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 1; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'I': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 1; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'J': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 1;  //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'K': {
        state[0] = 1; //1st servo
        state[1] = 0;
```

```
        state[2] = 0;
        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;
        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;
        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'L': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;
        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;
        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;
        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'M': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;
        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;
        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;
        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'N': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;
        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;
        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0;
        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'O': {
        state[0] = 1;  //1st servo
        state[1] = 0;
        state[2] = 0;
        state[3] = 0;  //2nd servo
        state[4] = 1;
        state[5] = 0;
        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;
        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'P': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;
        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;
        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;
        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'Q': {
        state[0] = 0;  //1st servo
        state[1] = 1;
        state[2] = 0;
        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;
        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;
        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'R': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;
        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;
        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;
        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'S': {
        state[0] = 0;  //1st servo
        state[1] = 0;
        state[2] = 1;
        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;
        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;
```

```
        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'T': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0;  //2nd servo
        state[4] = 1;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case 'U': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case 'V': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case 'W': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 1; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0;  //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case 'X': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case 'Y': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0;  //2nd servo
        state[4] = 1;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case 'Z': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0;  //2nd servo
        state[4] = 1;
        state[5] = 0;

        state[6] = 0;  //3rd servo
        state[7] = 0;
        state[8] = 1 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    //NUMBERS
    case '0': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case '1': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
```

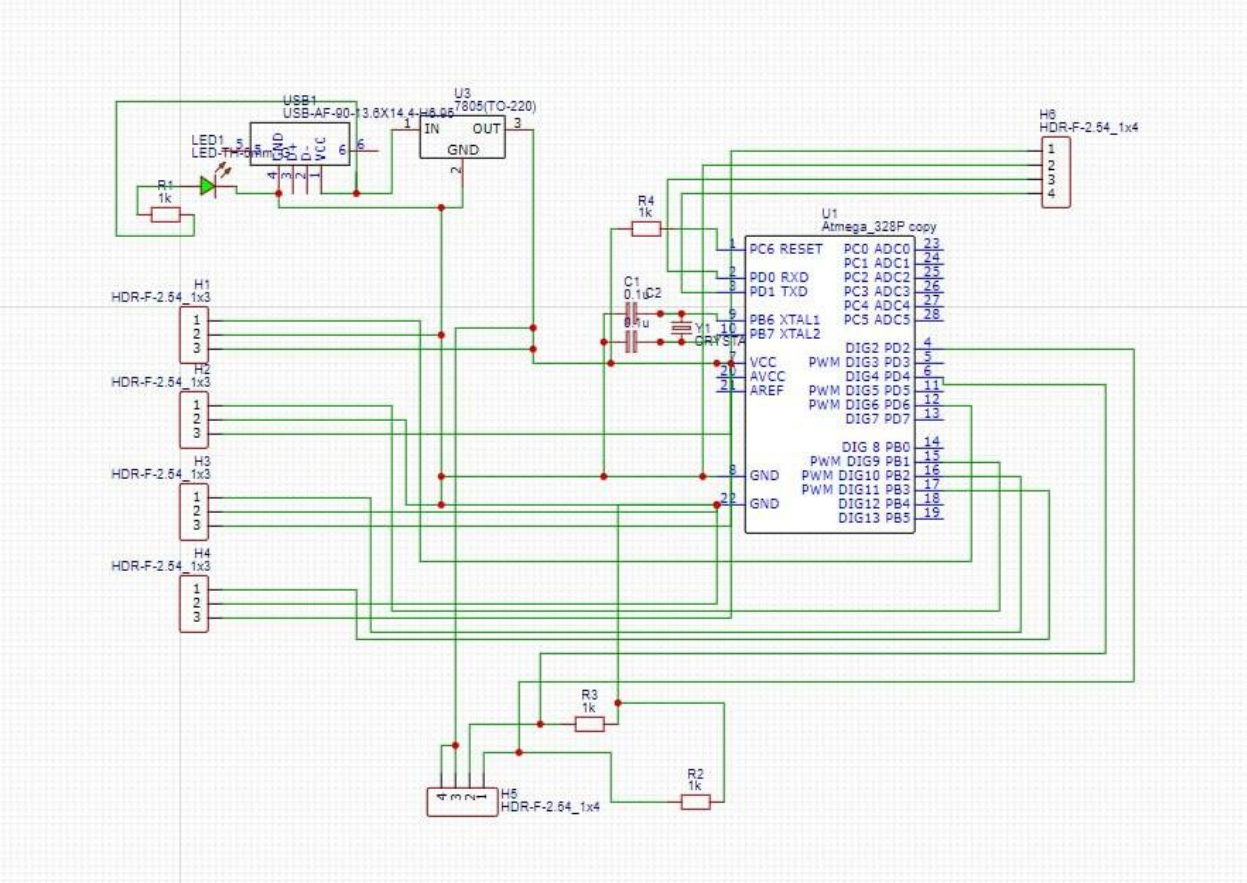
```
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case '2': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case '3': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case '4': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case '5': {
        state[0] = 1; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case '6': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case '7': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case '8': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;

        break;
      }
    case '9': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 1; //3rd servo
        state[7] = 0;
        state[8] = 0;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    //Punctuation
    case '!': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0; //2nd servo
```

```
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case '(': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case ')': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case '*': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case ',': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case '-': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case '.': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case ':': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 1 ;

        state[9] = 0;  //4th servo4
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case ';': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 1;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case '?': {
        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 1;
```

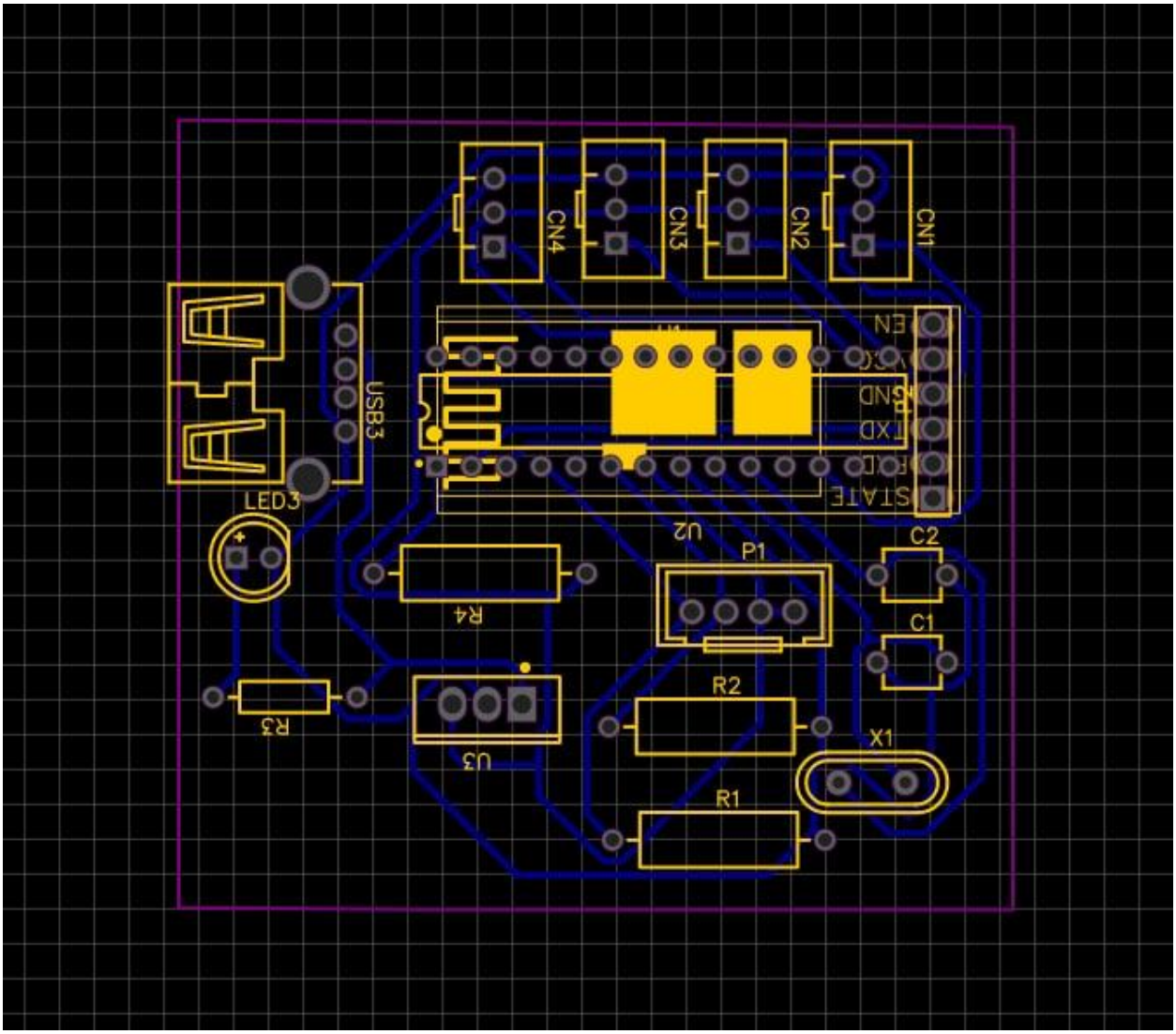
```
        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0 ;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 1;
        break;
      }
    case ' ': {

        state[0] = 0; //1st servo
        state[1] = 0;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 0;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 0;
        state[8] = 0;

        state[9] = 0; //4th servo
        state[10] = 0;
        state[11] = 0;
        break;
      }
    case '$': {
        state[0] = 0; //1st servo
        state[1] = 1;
        state[2] = 0;

        state[3] = 0; //2nd servo
        state[4] = 1;
        state[5] = 0;

        state[6] = 0; //3rd servo
        state[7] = 1;
        state[8] = 0;

        state[9] = 0; //4th servo
        state[10] = 1;
        state[11] = 0;
        break;
      }
  }
}
/*Displaying character which is
  converted to braille system*/
void displayChar() {
  //First Servo
  if (state[0] || state[1] || state[2])
  {
    //1st case
    if (state[0])
    {
      ser1.write(65);
    }
    //2nd case
    else if (state[1])
    {
      ser1.write(95);
    }
    //3rd case
    else if (state[2])
    {
      ser1.write(135);
    }
  } else
  {
    //4th case
    ser1.write(20);
  }
  //Second Servo
  if (state[3] || state[4] || state[5])
  {
    //1st case
    if (state[3])
    {
      ser2.write(5);
    }
    //2nd case
    else if (state[4])
    {
      ser2.write(48);
    }
    //3rd case
    else if (state[5])
    {
      ser2.write(85);
    }
  } else
  {
    //4th case
    ser2.write(170);
  }
  //Third Servo
  if (state[6] || state[7] || state[8])
  {
    //1st case
    if (state[6])
    {
      ser3.write(0);
    }
    //2nd case
    else if (state[7])
    {
      ser3.write(40);
    }
    //3rd case
    else if (state[8])
    {
      ser3.write(70);
    }
  } else
  {
    //4th case
    ser3.write(150);
  }
  //Fourth Servo
  if (state[9] || state[10] || state[11])
  {
    //1st case
    if (state[9])
    {
      ser4.write(35);
    }
    //2nd case
    else if (state[10])
    {
      ser4.write(75);
    }
    //3rd case
    else if (state[11])
    {
      ser4.write(115);
    }
  } else
  {
    //4th case
    ser4.write(0);
  }
  delay(100);
}
```

Appendix 6.

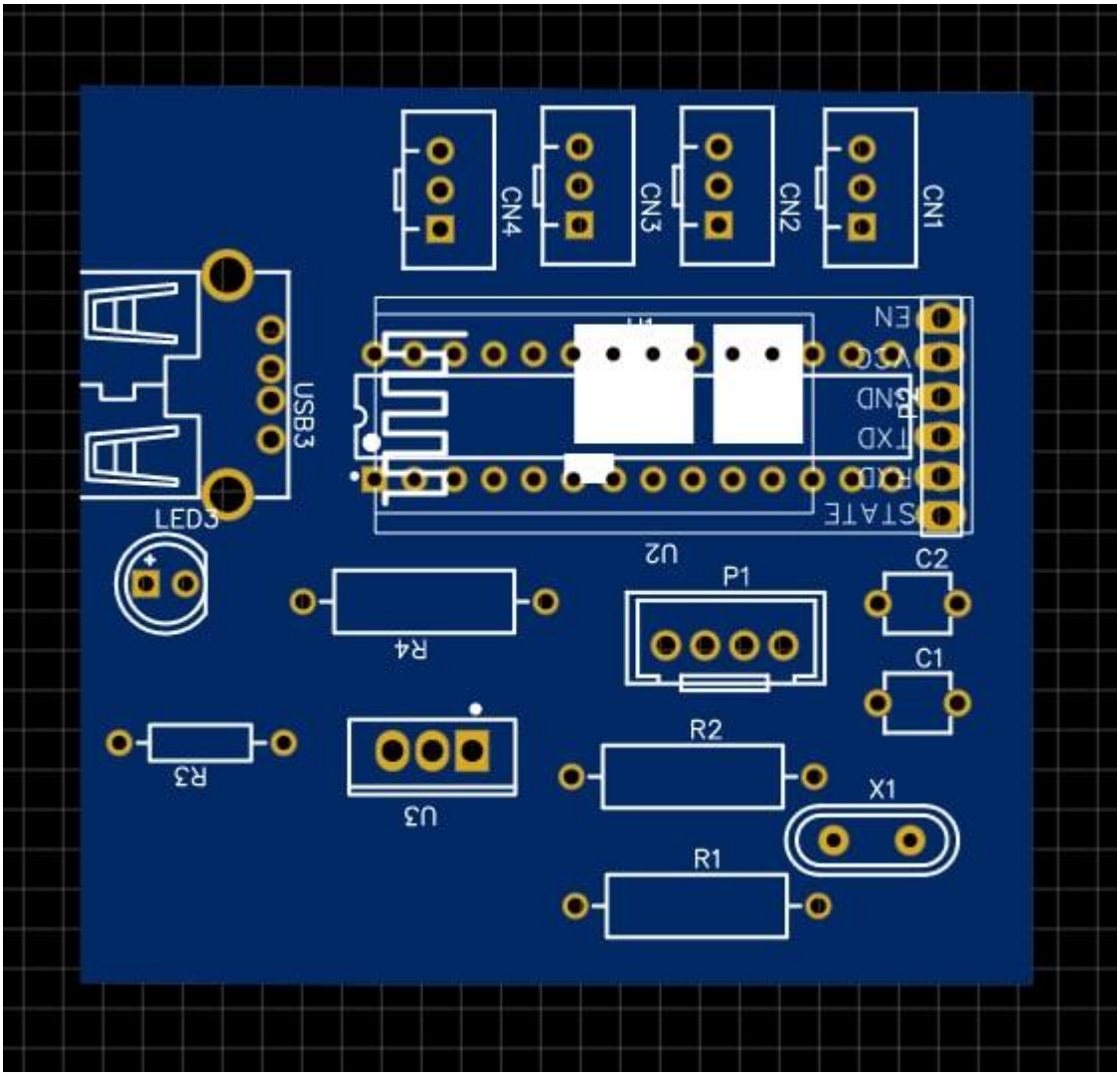
the schematics used in controlling the servo motors with Atmega328 microcontroller.



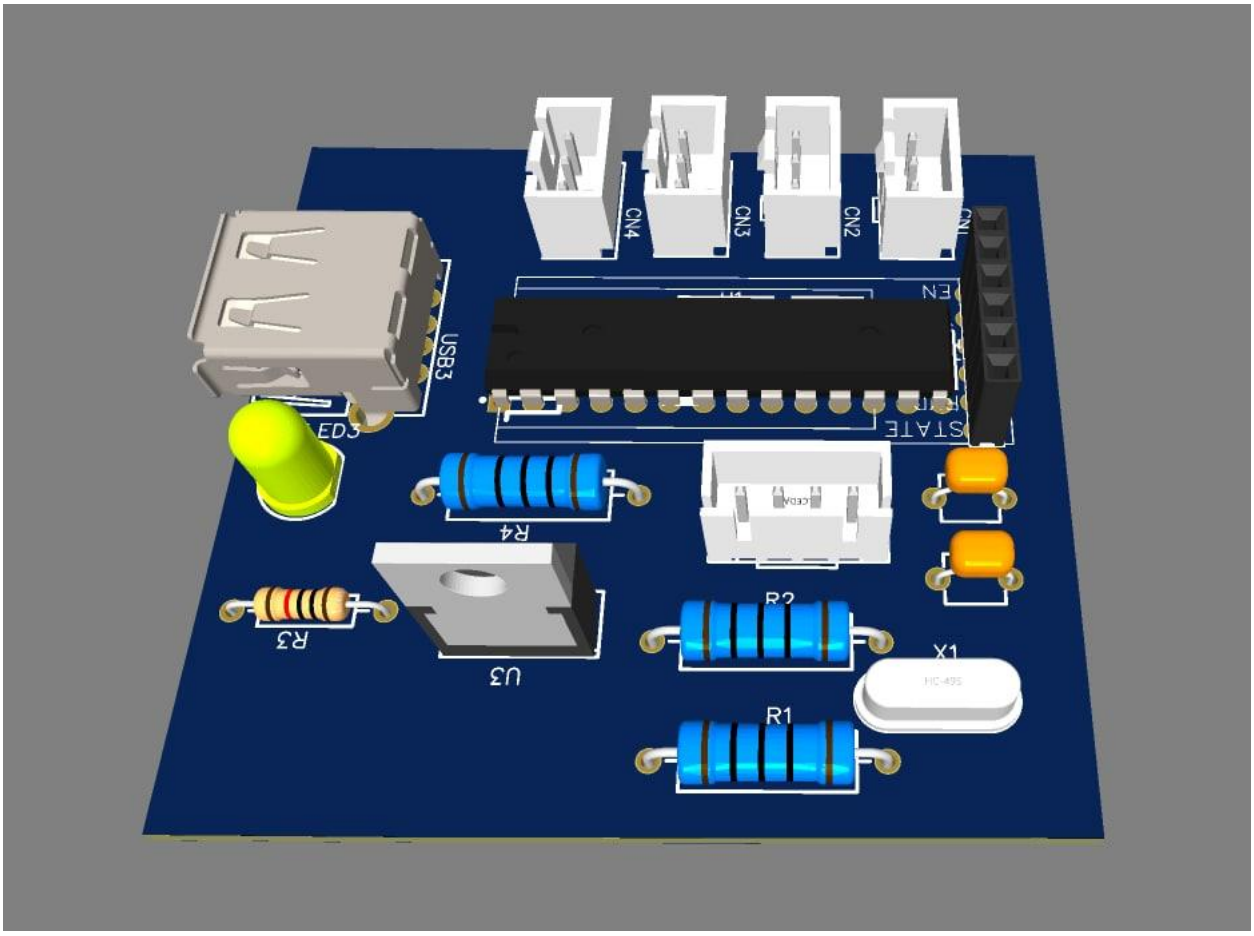
To find optimal place for Placing the components and routing lines according to main schematics



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