# Princess Sumaya University for Technology

## King Abdullah II Faculty of Engineering

## Computer Engineering Department


Princess Sumaya جامعـــة
University الأميـــرة سميّـــة
for Technology للتكنولوجيا

# BRAILLE-TO-ENGLISH CNC MACHINE

## EMBEDDED SYSTEMS FINAL PROJECT


***Authors:***

| | | |
|---|---|---|
| **Mahmoud Abu-Qtiesh** | 20210383 | Computer Engineering |
| **Ghaith Abboud** | 20210055 | NIS Engineering |
| **Abdulkarim Damisi** | 20210173 | Communication/IOT Engineering |

**Abstract**

This project focuses on developing an innovative Braille-to-English CNC machine using the PIC16F877A microcontroller. The machine is designed to interpret 6-bit Braille input and translate it into English letters, which are then drawn on a surface using precise motor controls. It features dual functionalities: real-time LCD feedback for displaying the translated letters and a dynamic size adjustment mechanism controlled by an analog input. The machine operates two stepper motors for positioning and a servomotor for pen control, ensuring accuracy and efficiency. Inspired by the Perkins Brailler, the system combines assistive technology with CNC capabilities to enhance accessibility for visually impaired individuals. This comprehensive report delves into the project's hardware, mechanical and software design, addresses challenges faced during development, and highlights creative solutions. By integrating robotics, embedded systems, and accessibility technologies, this project demonstrates technical innovation and contributes to advancing assistive device technology.

# TABLE OF CONTENTS

1 Introduction ..... 2

2 Software Design ..... 3

2.1 Software Libraries ..... 3

2.1.1 Draw Base Library ..... 3

2.1.2 Draw Letters ..... 4

2.1.3 Analog-to-Digital ..... 5

2.1.4 Timer Initialization ..... 6

2.2 Braille Input ..... 6

2.3 Software Architecture ..... 7

# 1 INTRODUCTION

# 2 SOFTWARE DESIGN

This section of the report outlines the software engineering processes involved in designing the system as a whole. The diagram below provides a high-level overview of the software system's architecture and design.

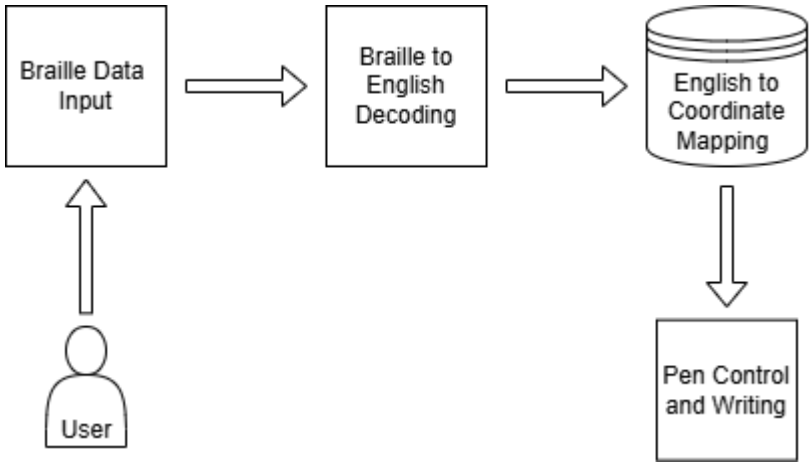



Figure 1: Braille-to-English System Diagram

The process begins with the user entering a Braille input that corresponds to the letter they wish to write. This input is then decoded into its equivalent English letter. Next, the system determines the precise movements required to write the letter and executes them to produce the letter on the paper.

## SOFTWARE LIBRARIES

For this project, the use of external libraries, apart from the LCD library, was prohibited. As a result, all other libraries were custom-written specifically for this project, providing greater control over low-level functionalities such as PWM signal generation and coordinate mapping.

### 2.1.1 Draw Base Library

One of the initial challenges encountered during development was mapping English letters to coordinates, as illustrated in Figure 1. Ideally, a universal G-code receiver could have been used to interpret the G-code generated from the input. However, this approach was not feasible due to the limitations of using a PIC microcontroller. Another potential solution involved using a "7-segment"-style display, but this would have resulted in letters that were difficult to interpret. Finally, the chosen solution was to implement a 20-segment grid, tailor-made for the project to ensure clarity and accuracy.

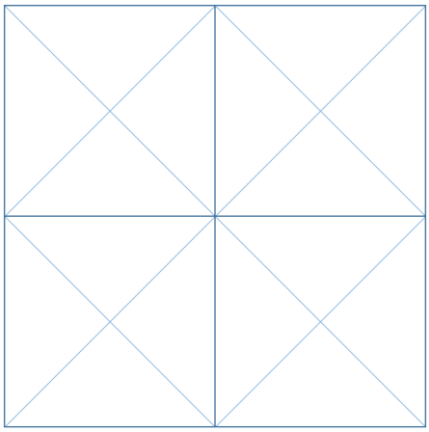


Figure 2: 20-Segment Grid

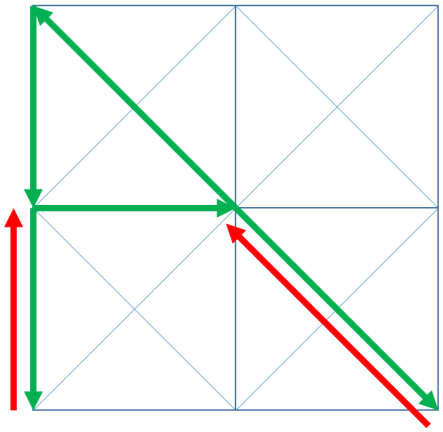


Figure 3: Writing "A" using 20-segment Grid

The Draw Base library defines eight functions that control the pen's movements. These functions include `draw_right(size)`, `draw_up(size)`, and `draw_up_right(size)`, which send signals to move the pen to the right, upward, and diagonally upward to the right, respectively.

### 2.1.2 Draw Letters

Due to linking issues, the Draw Letters library was integrated into the project's main file instead of being placed in a separate header file. Despite this change, the Draw Letters functions utilize the methods defined in the Draw Base library to write letters on the 20-segment grid.

When writing a letter, the pen always starts at the origin of the grid as can be seen from Figure 3, located at the intersection of the four central sub-squares. From there, the pen can move in any direction, ensuring it remains within the grid's boundaries. Once the writing process is complete, the function ensures the pen returns to the grid's origin, preparing it for the next letter to be written seamlessly.

The library defines, 26 functions for each character alongside to 5 additional utility functions, such as `pen_down()` and `pen_up()`, which control the pen's position by lowering and raising it. For instance, the red arrows in Figure 3, indicate movements where the pen wasn't touching the paper. In addition function `move_letter()` repositions the pen to the starting point of the next letter, `draw_space()` which draws an empty letter and finally `enter_new_line()` moves the pen to the start of the next line.

Figure 4 below shows a simple "Hello World" program that was used for testing due to its utilization of all eight Draw Base functions. In addition to this test, there were many extensive tests for each character to ensure that every letter is written as intended.

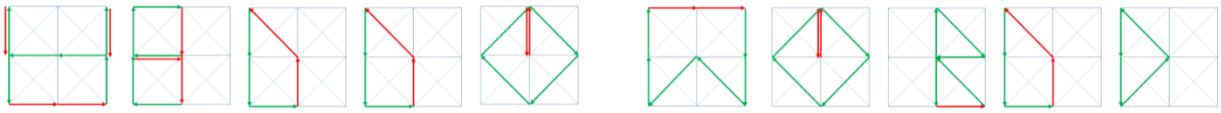


Figure 4: Hello World Program

### 2.1.3 Analog-to-Digital

To meet the requirement of including an analog input, a potentiometer was incorporated to control the size of the letters. Initially, the letter size was fixed, with a delay of 100ms used to ensure each segment of a letter was properly drawn. This delay has since been made adjustable, ranging from 50ms to 150ms, providing greater control over the drawing speed.

The ATD (Analog-to-Digital) library provides two key functions: `ATD_Init(),` which initializes the analog-to-digital conversion, and `ATD_Read(port),` which reads the value from the potentiometer. The raw potentiometer value ranges from 0 to 255. To map this range to the desired delay range of 50ms to 150ms, the following function is applied:

$$size = 50 + \left(\frac{ATD_{read} \cdot (150 - 50)}{255}\right)$$

The adjustable letter size introduces a challenge in the design, particularly in ensuring non-blocking execution. Larger letters significantly increase the time required to draw each character. For example, the letter 'H' takes 750ms to draw at the smallest size as can be seen in Figure 5, which is among the longest durations for any character. At the largest size, this time increases to 1850ms, meaning users would experience longer wait times when writing larger letters. As a result, optimizing the design to handle these delays efficiently is crucial to maintain a smooth user experience.


Figure 5: Letter Size Timing Difference Ordered from Longest to Shortest

To address this issue, instead of modifying the delay for the stepper motor, which operates at a constant speed, we keep the delay fixed and adjust the motor's speed. While the distance covered by the stepper changes in both approaches, the second method significantly reduces the overall time required.

This was accomplished by dynamically adjusting the reload value of TMR0, which generates the PWM signal for both stepper motors. The adjustment was implemented using the following function:

$$\text{TMR0} = 0\text{xE8} + \frac{(\text{size} - 50) \cdot (0\text{xF8} - 0\text{xE8})}{255 - 50}$$

Using this method, a letter with a size of 50 would cause the stepper motors to operate at a frequency of 5,208 Hz, while a letter with a size of 150 would increase the frequency to 15,625 Hz.

### 2.1.4 Timer Initialization

As explained above TMR0 is used to generate the PWM signal for both stepper motors allowing them to move at the same pace. The servomotor is controlled using TMR1 and the CCP module to generate precise PWM signals. The high pulse width of the signal determines the servo's position, with the angle variable dynamically adjusting this width. The system ensures that each signal cycle, typically 20ms, includes both a high and low pulse, allowing the servo to move accurately to the desired angle. This method provides efficient and reliable control over the servo motor's positioning.

## BRAILLE INPUT

Before delving into the mechanics of how the braille inputs works, it is important to explain the structure of braille letters.

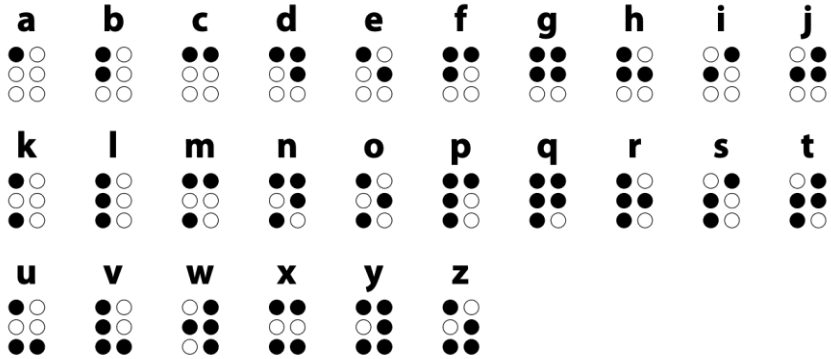



Figure 6: Braille Alphabet

Each character in Braille, known as a "cell," is made up of six dots arranged in a rectangular grid of two columns and three rows. The dots are numbered from 1 to 6, starting from the top-left and moving to the left.

Different combinations of raised and unraised dots within a cell represent letters, numbers, punctuation, and even entire words in some cases. For example, in Figure 6 the letter "A" is represented by filling only dot 1, while the letter "B" is represented by filling dots 1 and 3.

To account for the 6 dots in a Braille cell, 6 buttons were implemented to represent each dot. Additionally, an "Enter" button was included to initiate the writing of the corresponding letter on the paper, along with a "Clear" button to reset any selected dots, allowing for corrections before finalizing the input. The braille keyboard layout will be explained further in the hardware design section.

## SOFTWARE ARCHITECTURE

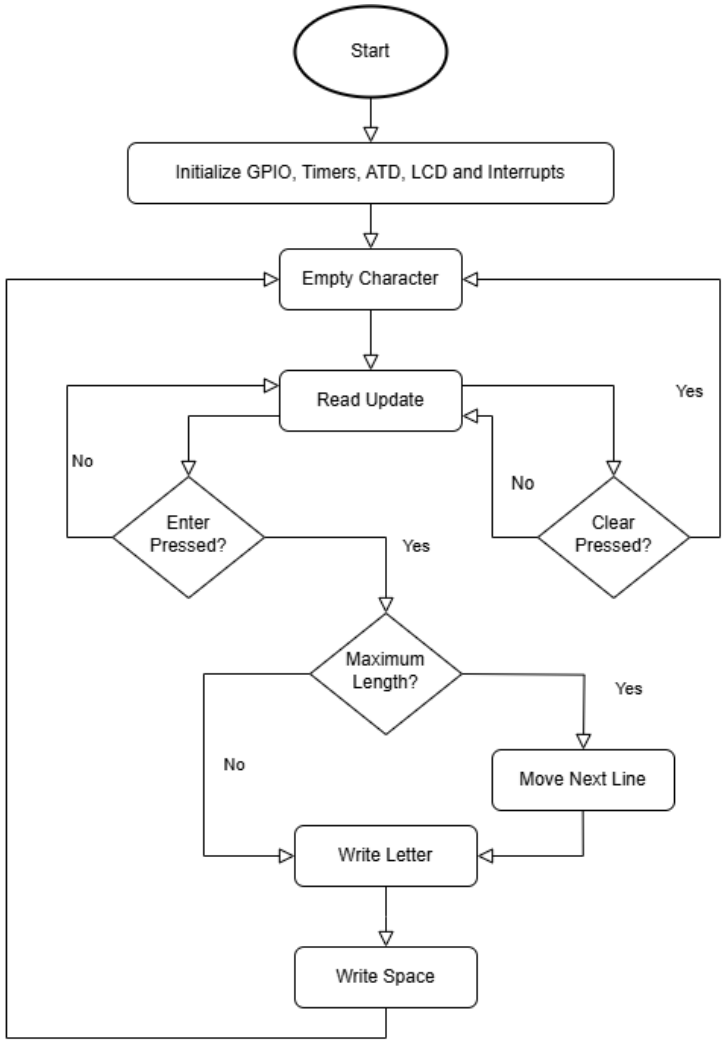



Figure 7: Braille-to-English CNC Machine Flowchart

This flowchart provides an overview of the system's operation for processing input and generating output. It starts with initializing the system's hardware components, such as GPIO, timers, LCD, and interrupts. The system then continuously monitors for updates and checks for specific actions, such as clearing the input or confirming it with an "Enter" command. If the input is valid and within the allowed length, it processes the data by writing letters and spaces. When the input exceeds the maximum length, the system moves to the next line and continues processing. The flow ensures smooth input handling, writing, and formatting in an iterative and logical manner.

# 3 HARDWARE DESIGN

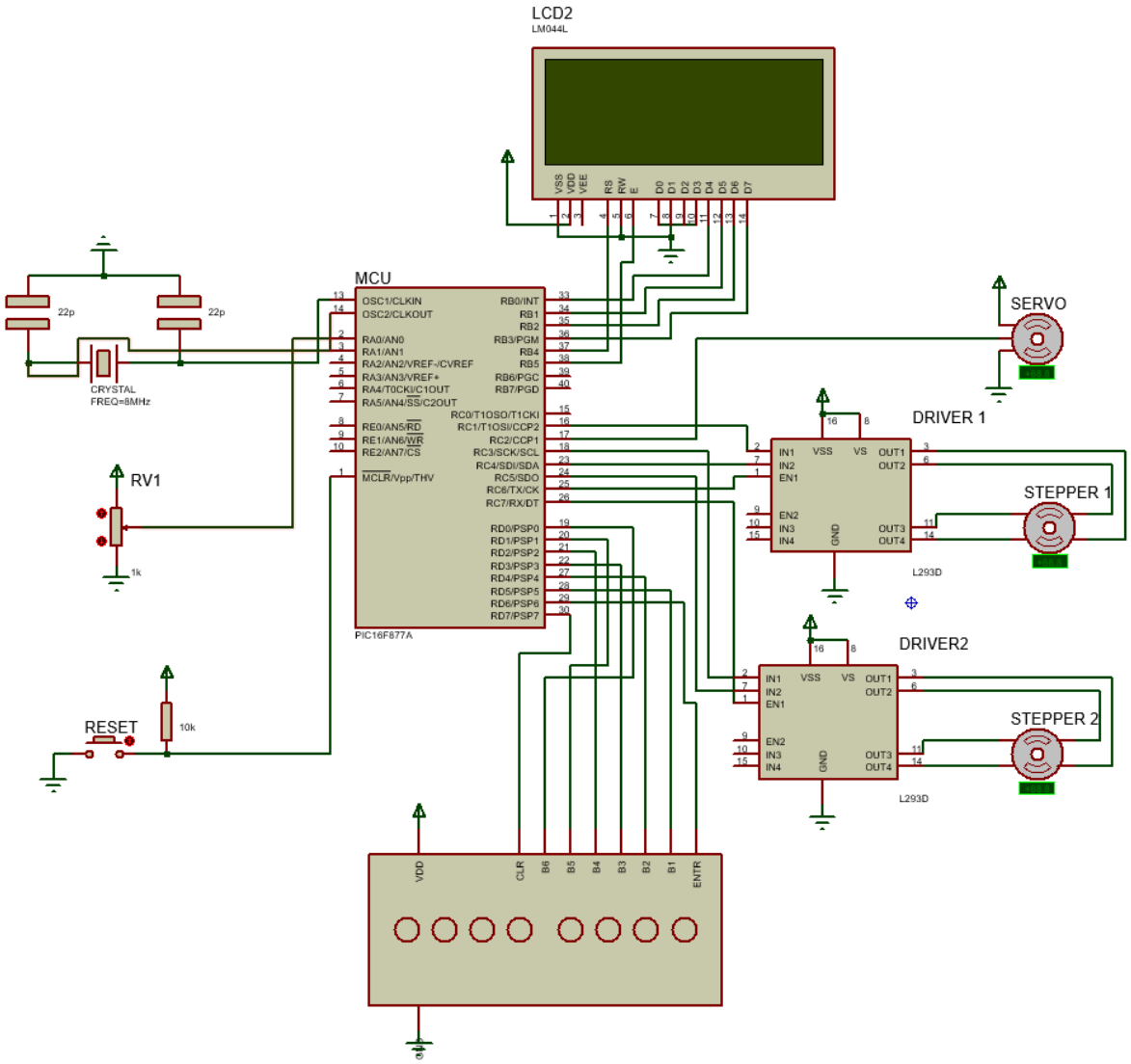



Figure 8: Braille-to-English CNC Machine Schematic

Figure 8 illustrates the complete circuit, with the PIC16F877A microcontroller at its core, operating at an 8 MHz clock frequency. The stepper motors and the servo motor are connected to PORTC, with the motors controlled via motor drivers and the servomotor controlled using PWM signals through a CCP pin. The Braille keyboard, consisting of six buttons for the Braille dots along with "Enter" and "Clear" buttons, is connected to PORTD for input. The LCD screen, used for real-time feedback such as letter size and system status, is connected to PORTB. Additionally, a potentiometer on the RA0 pin adjusts the letter size dynamically, and a reset button is included for system reinitialization. This design ensures smooth integration and precise control of all components.

## STEPPER MOTORS

Figure 9: Nema 17 Stepper Motor

The stepper motor was interfaced with the PIC microcontroller using an A4988 driver and its shield. The driver was connected to an expansion board with a current-limiting capacitor for stable operation. The reference voltage $V_{\text{ref}}$ was adjusted using an equation to set the current limits, ensuring reliable and efficient motor control.

$$V_{\text{ref}} = 1.3\ \text{A} \cdot 8 \cdot 0.1\ \Omega$$

## SERVO MOTOR


Figure 10: MG90s Servo Motor

The MG90S servo motor was used to raise and lower the pen, allowing the CNC machine to transition from 2D to 2.5D operation. The control signal for the servo was generated using Timer1 and the CCP module in compare mode, with the angle determining the pulse width. This ensured precise timing for the high and low pulses, allowing accurate control of the servo's position.

## LCD DISPLAY

For debugging and testing an LCD screen was used in junction with the pen to ensure that input was correct and that size manipulation was also correct it also shows the output that is written onto the paper.

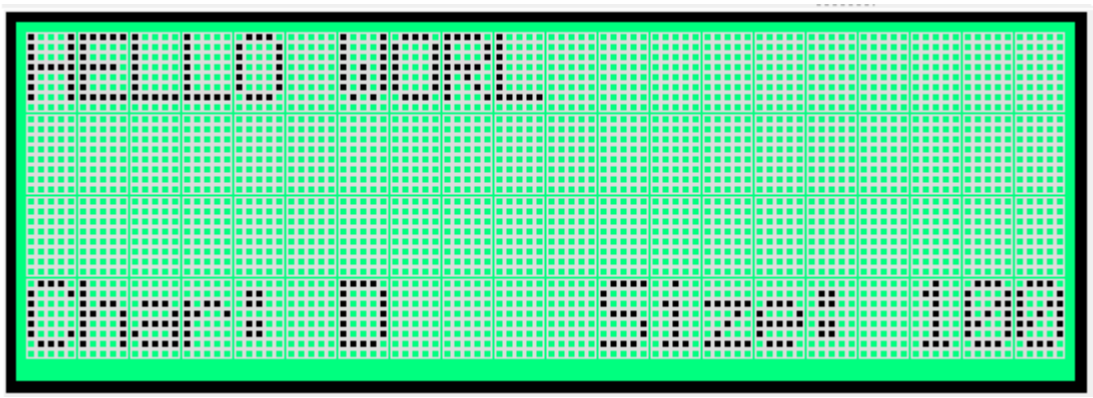
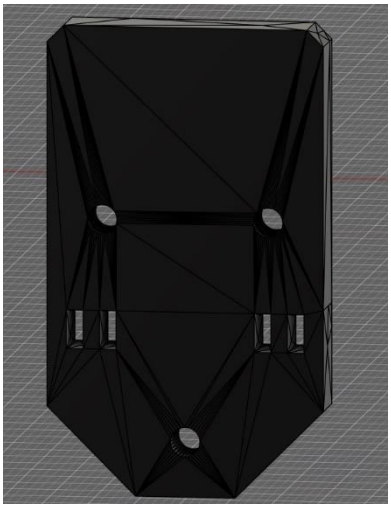
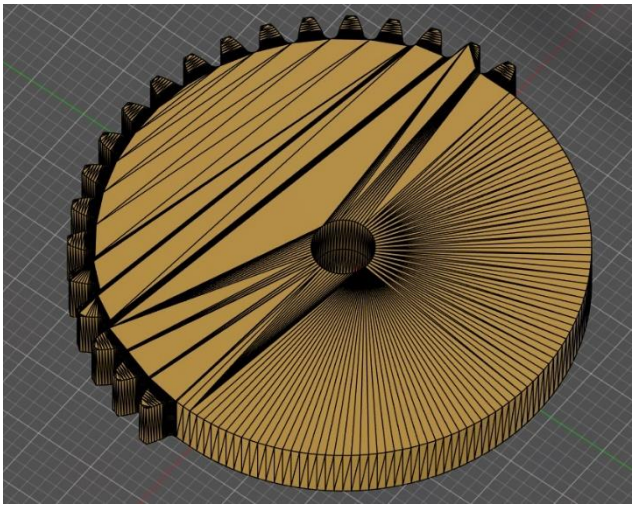


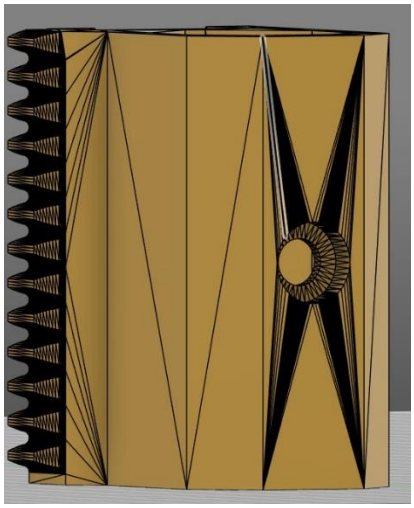
Figure 11: LCD Display Layout

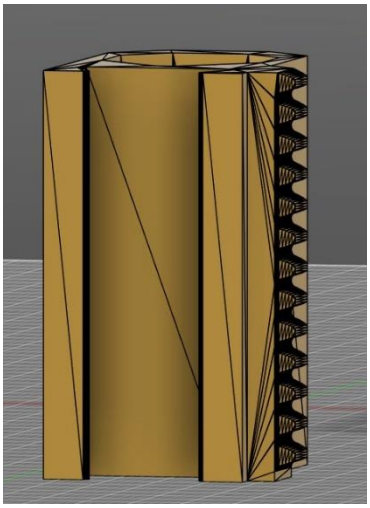
When the current line fills up the cursor moves to the next line in addition the stepper motors move to the line underneath. The fourth line shows the current character as seen in Figure 11: LCD Display LayoutFigure 11 in addition to the size.

# 4 MECHANICAL DESIGN









The mechanical design of the system combines 3D-printed components and aluminum profiles to achieve precise and stable movement of the pen. The aluminum profiles serve as the primary structural framework, providing rigidity and durability to support the stepper motors and the moving components. These profiles act as linear guides along which the pen assembly travels, ensuring smooth and accurate motion during operation.

The 3D-printed parts play a crucial role in housing and connecting the motors, gears, and other mechanical components. These custom-printed parts are designed to fit seamlessly with the aluminum profiles and the stepper motors, allowing efficient transfer of motion. Additionally, the 3D-printed mounts and brackets secure the pen in place while enabling easy adjustments and alignment. This combination of lightweight, customizable 3D-printed parts and robust aluminum profiles ensures a cost-effective yet highly functional design, capable of precise and repeatable movements for writing and drawing task

# 5 RESULTS AND DISCUSSION

After connecting all the components we achiei

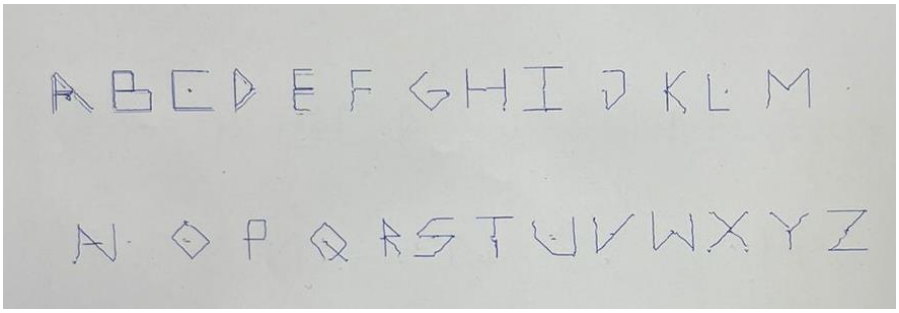


Figure 12: English Alphabet using CNC machine

As can be seen from figure 12 the English alphabet is very clear compared to the expected results future work would involve adding Arabic letters, increasing the 20-segment grid to an 80 segment grid allowing us to draw letters more clearly.

# 6 CONCLUSION

In conclusion, this project successfully integrates mechanical, electrical, and software components to design and implement a Braille-to-English CNC machine. The use of a PIC16F877A microcontroller, stepper motors, a servo motor, and an LCD display demonstrates the seamless combination of hardware and software to achieve precise and reliable operation. The mechanical framework, comprising aluminum profiles and 3D-printed parts, ensures structural stability and smooth motion, while the software, including custom libraries and dynamic control algorithms, allows for flexibility and accurate functionality. The incorporation of user-friendly features, such as the Braille keyboard and adjustable letter size, highlights the practical usability of the system. This project not only meets the initial design goals but also provides a solid foundation for future enhancements and applications in assistive technologies.