INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

A Project Report

On

Transforming Alphabets into Touch:
A Braille Language Project

⠠⠁⠀⠠⠏⠗⠕⠚⠑⠉⠞⠀⠠⠗⠑⠏⠕⠗⠞⠀⠕⠝
⠠⠁⠀⠠⠃⠗⠁⠊⠇⠇⠑⠀⠠⠇⠁⠝⠛⠥⠁⠛⠑⠀⠠⠏⠗⠕⠚⠑⠉⠞

By:

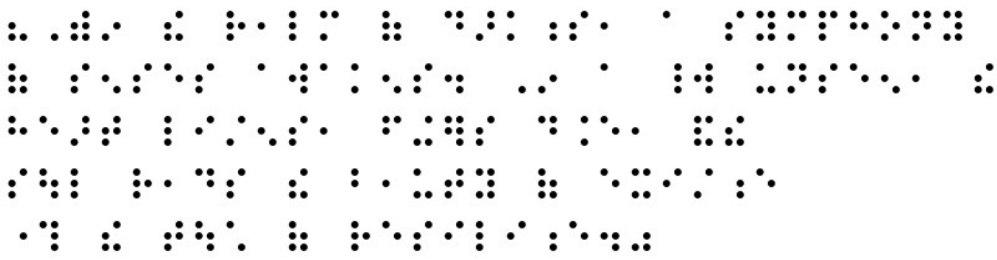
Yugesh Bhoge (23N0278)
Toshan Kumar Dhaker (23N0281)
Manish Varshney (23N0289)
Mainak Nandi (23N0325)

Guided By:

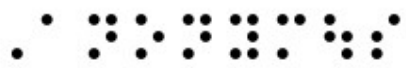
Prof. Tapanendu Kundu

"

Within the realm of darkness, a symphony of senses awakens. In a world unseen, the heart listens, fingers dance, and the soul reads the beauty of existence through the touch of resilience."



——— *Anonymous*



1 INTRODUCTION:

In a world where communication is primarily visual and auditory, we often overlook the needs of individuals with visual impairments. This project addresses the gap by delving into the realm of Braille language conversion. Imagine a world where the written word becomes a tangible experience, accessible to everyone.

This project implements the logical processing power of Arduino Uno and the Wokwi simulation platform to create a unique and innovative system. The goal is simple yet profound: to convert ordinary alphabetic characters into Braille patterns and represent them physically through a set of servo motors. Through this endeavor, we aim to make written information not just readable but touchable, opening up a new dimension of literacy for individuals who rely on Braille.

In the following report, we will explore the intricacies of our Braille Language Project, discussing the components used, the mapping of Braille patterns to alphabets, and the seamless integration of Arduino Uno and Wokwi for simulation. We will also share insights into the testing process, challenges encountered, and potential avenues for future enhancements. "Transforming Alphabets into Touch" is not just a project; it's a step towards a more inclusive and accessible world of communication

2 OBJECTIVE:

The objectives of the "Transforming Alphabets into Touch: A Braille Language Project" are:

1. **Braille Conversion:** Develop a system to convert alphabetic characters into corresponding Braille patterns.
2. **Hardware Integration:** Integrate servo motors controlled by an Arduino Uno to physically represent Braille patterns.
3. **Simulation Environment:** Utilize the Wokwi simulation platform for real-time interaction and testing of the Braille language conversion.
4. **User Interface:** Establish serial communication for user input, allowing users to input alphabetic characters through the Serial Monitor.
5. **Inclusivity and Accessibility:** Create a tactile representation of written information for individuals with visual impairments, promoting accessibility and inclusivity in communication.

3 METHODOLOGY:

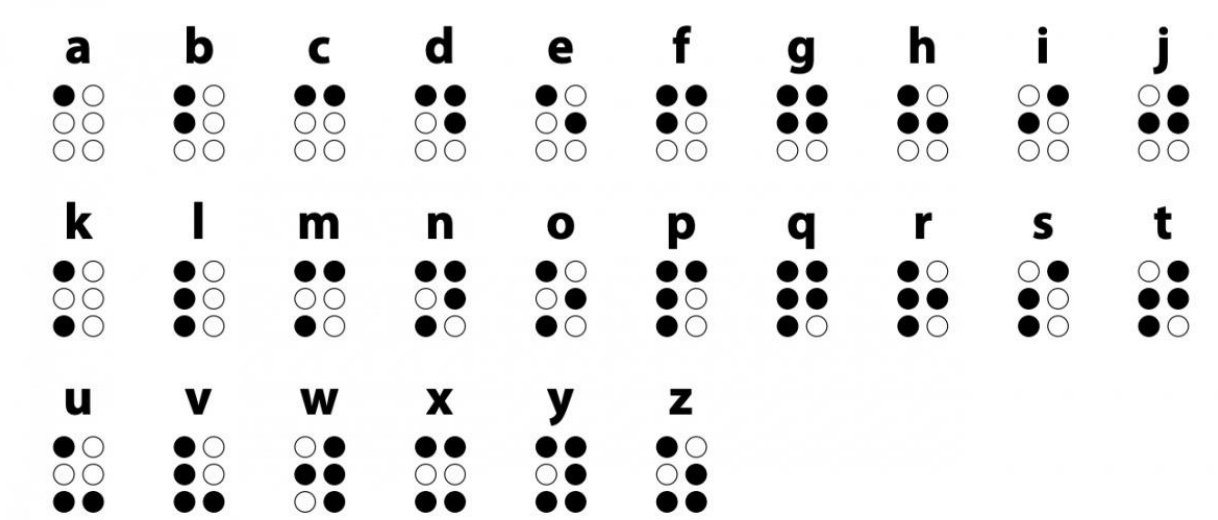
1. Project Planning and Definition:

- **Objective Clarity:**
  - Clearly defined the project objective: to convert alphabetic characters into Braille patterns for tactile representation using Arduino Uno and Wokwi.
- **Scope Definition:**
  - Outlined the scope of the project, including the use of servo motors for physical representation and integration with the Wokwi simulation platform.

2. Hardware and Software Selection:

- **Arduino Uno Selection:**
  - Chose Arduino Uno as the microcontroller for its compatibility, ease of use, and availability of servo control libraries.
- **Servo Motors:**
  - Selected servo motors for their precision and ability to control physical movements, crucial for representing Braille patterns.
- **Wokwi Simulation Platform:**
  - Opted for the Wokwi online simulation platform for its user-friendly interface and real-time interaction capabilities.

3. Braille Alphabet Mapping:



- **Mapping Design:**
  - Developed a comprehensive mapping of Braille patterns to alphabetic characters, ensuring accurate representation for all 26 letters.
- **Array Implementation:**
  - Translated the Braille mappings into a 2D boolean array in the Arduino code for efficient and organized representation.

4. Arduino Code Implementation:

- **Setup Function:**
  - Initialized serial communication and attached servo motors to designated pins.
- **Braille Conversion Functions:**
  - Implemented functions to convert alphabetic characters to their corresponding Braille patterns and control servo motors accordingly.
- **Serial Communication Handling:**
  - Developed a loop function to continuously monitor serial input, converting valid alphabets into Braille patterns and controlling motors.

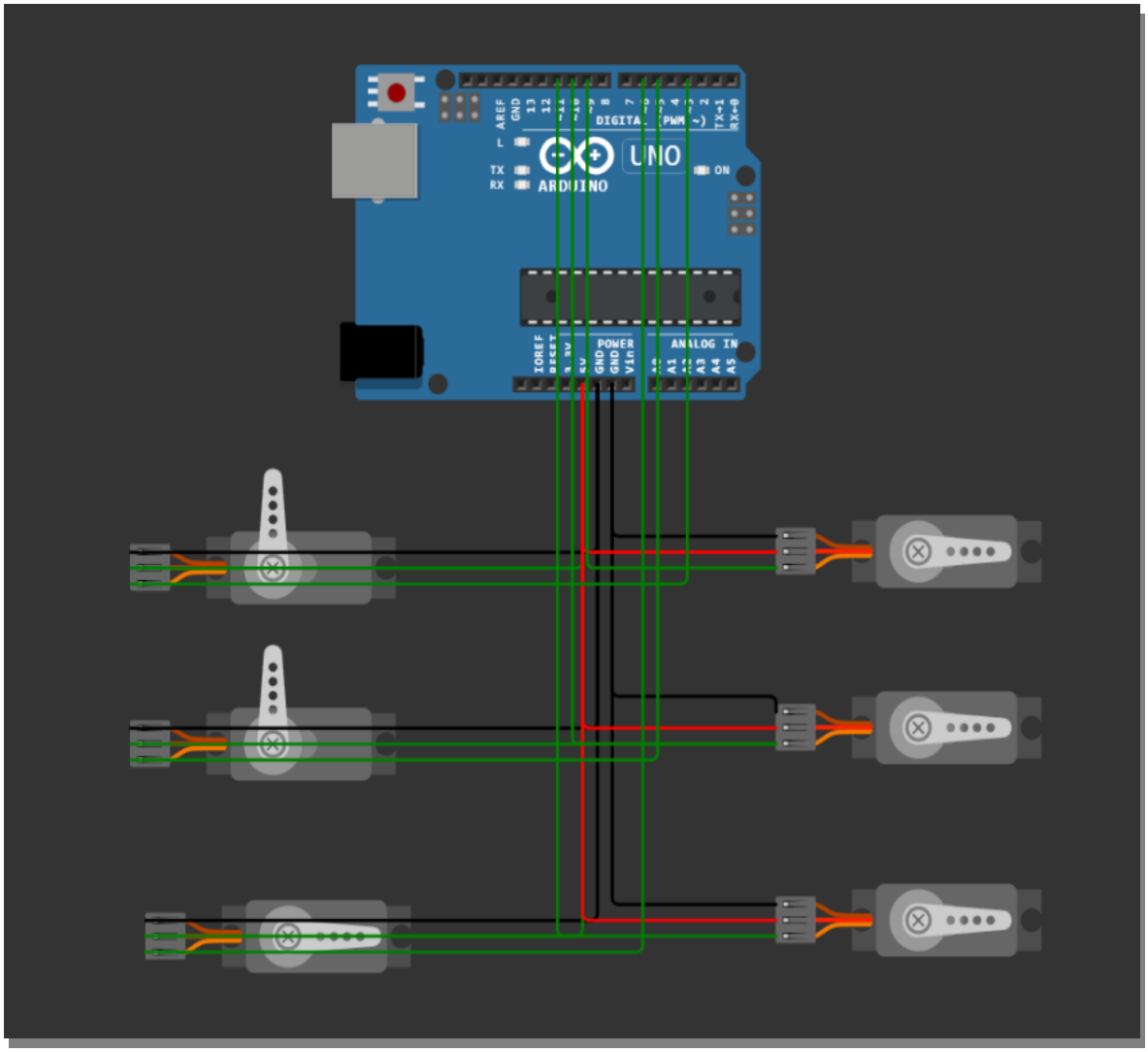
5. Wokwi Simulation Setup:

- **Circuit Design:**
  - Created a virtual circuit in Wokwi, including Arduino Uno and servo motors, ensuring accurate representation of the physical setup.
- **Serial Monitor Integration**:
  - Integrated the Wokwi Serial Monitor for real-time interaction, enabling character input and immediate visualization of the Braille output.

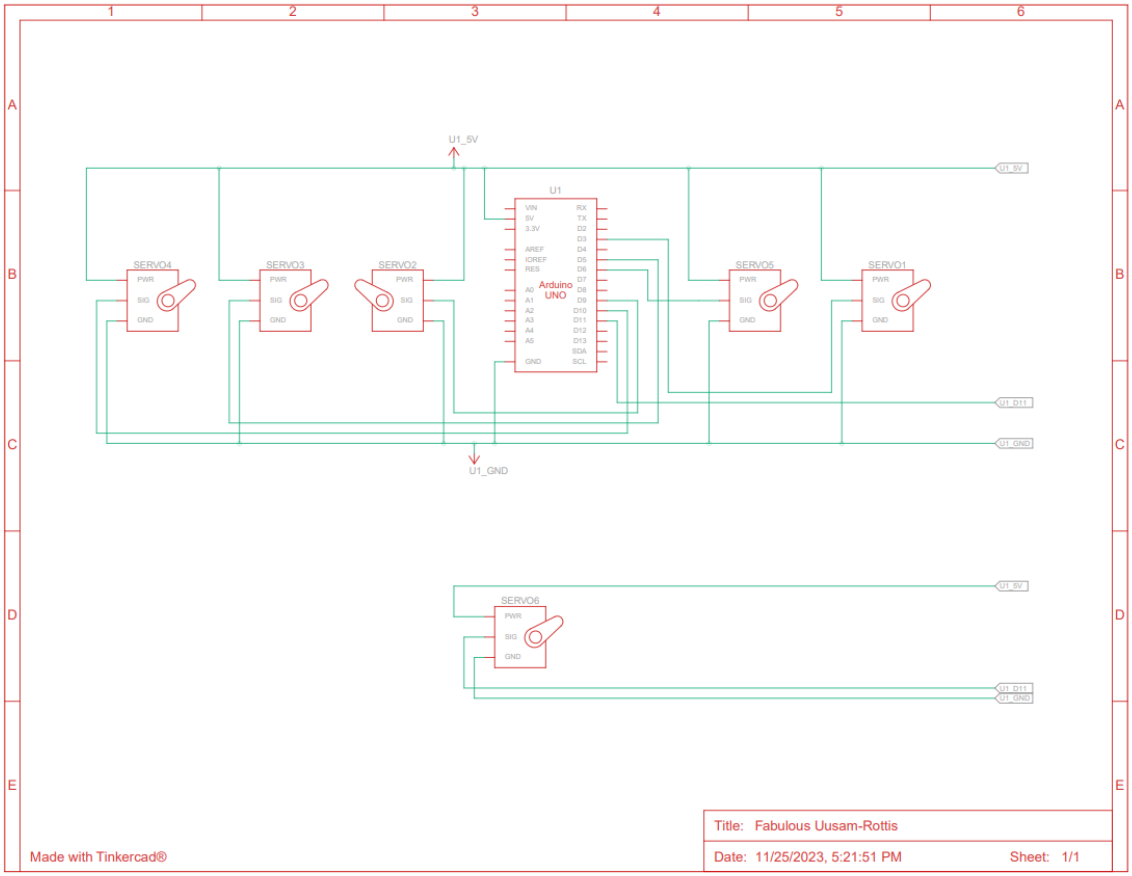
6. Testing and Debugging:

- **Serial Input Testing**:
  - Conducted thorough testing of serial input handling to ensure accurate reception and conversion of characters.
- **Motor Control Testing:**
  - Rigorously tested the servo motor control mechanism, verifying that Braille patterns were accurately represented.
- **Debugging:**
  - Addressed and resolved any issues through iterative debugging, refining the code for optimal performance.

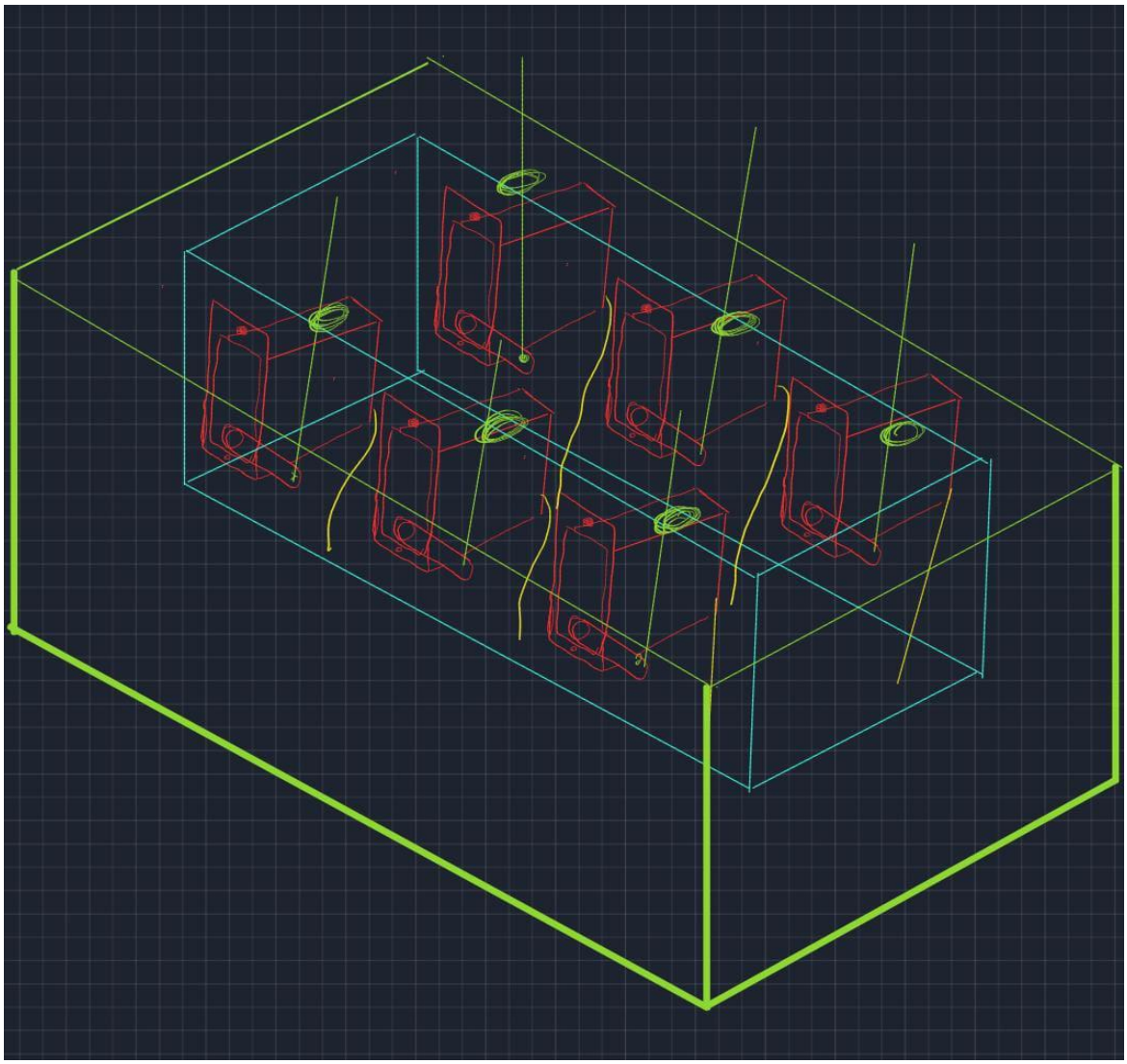
4 CIRCUIT DIAGRAM:



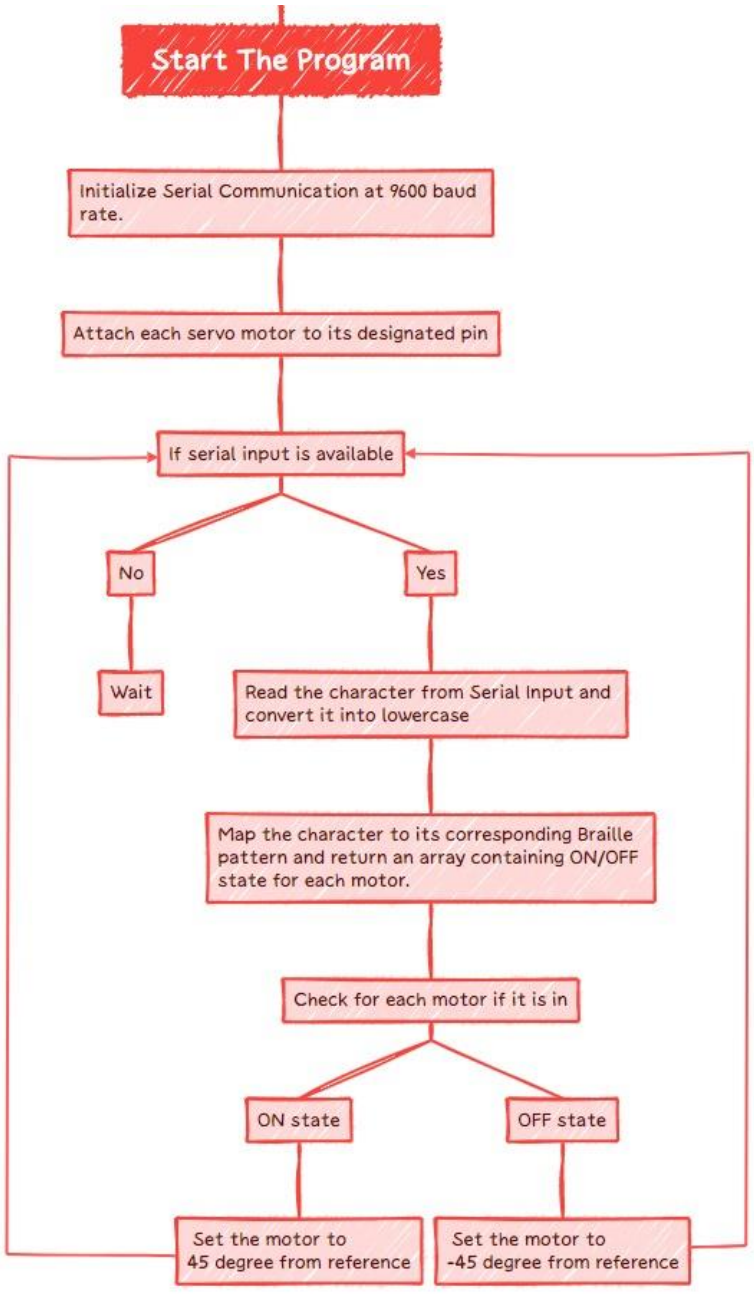
5 SCHEMATIC DIAGRAM:



6 DESIGN DRAWING:



7 FLOW CHART:


Start The Program
Initialize Serial Communication at 9600 baud rate.
Attach each servo motor to its designated pin
If serial input is available
No
Yes
Wait
Read the character from Serial Input and convert it into lowercase
Map the character to its corresponding Braille pattern and return an array containing ON/OFF state for each motor.
Check for each motor if it is in
ON state
OFF state
Set the motor to 45 degree from reference
Set the motor to -45 degree from reference

8 EXPLANATION OF ALGORITHAM:

1. Define Motor Control Constants:

- Sets up an array `motorPins` to store the pin numbers for six servo motors.
- Declares an array of Servo objects named motors to control the servo motors.

2. Braille Alphabet Mapping:

- Defines a 2D boolean array `brailleAlphabet` to store Braille patterns for each letter of the alphabet.

3. Setup Function:

- Initializes serial communication at a baud rate of 9600.
- Prints a welcome message to the Serial Monitor.
- Attaches each servo motor to its respective pin.

4. Loop Function:

- Checks if there's data available in the Serial input buffer.
- Reads a character from the Serial input.
- Checks if the input is an alphabet character.
- Converts the alphabet character to its corresponding Braille pattern.
- Controls the servo motors based on the Braille pattern.
- Includes a small delay to prevent locking up.

7. Braille Conversion Functions:

a. `charToBraille`: Converts an alphabet character to its corresponding Braille pattern.

b. `controlMotors`: Controls the servo motors based on the Braille pattern. If a dot is present, the motor is set to write 0; otherwise, it's set to write 90.

This code essentially sets up a system where you can input alphabet characters through the Serial Monitor, and the corresponding Braille pattern is represented by the servo motors. The Braille dots are simulated by the position of the servo motors, creating a tactile representation of the input alphabet.

9 RESULT:

The results of the "Transforming Alphabets into Touch: A Braille Language Project" are as follows:

1. **Successful Braille Conversion:** The project effectively converts alphabetic characters into Braille patterns, enabling a tactile representation of written information.
2. **Accurate Motor Control:** Servo motors controlled by the Arduino Uno accurately simulate Braille dots, providing a physical representation that users can feel.
3. **Real-time Interaction in Simulation:** The integration with the Wokwi simulation platform allows for real-time interaction and testing of the Braille language conversion, enhancing the project's user-friendliness.
4. **User Input Handling:** The system efficiently handles user input through the Serial Monitor, allowing users to input alphabetic characters and observe the corresponding Braille output.
5. **Enhanced Accessibility:** The project achieves its goal of making written information accessible to individuals with visual impairments by offering a touchable and tangible representation of the alphabet.

10 CONCLUSION:

In conclusion, our Braille Language Project successfully turns regular letters into touchable Braille patterns using Arduino and Wokwi. The project allows people with visual impairments to feel and understand the alphabet. The motors move to create Braille dots that represent each letter. Through the Wokwi simulation, we made sure the system works well and is easy to use.

This project is a step towards making information accessible to everyone, regardless of their abilities. We are happy with the results and look forward to making it even better in the future by adding more features. Overall, it's a small project with a big impact, making written language something you can touch and feel.

11 FUTURE SCOPE:

More Characters: We can expand the system to include numbers and special characters, making it useful for a wider range of information.

a. **Enhanced Interaction:** Adding buttons or sensors could allow users to interact with the system more easily, making it even more user-friendly.
b. **Educational Tools:** The project can be extended to become a helpful educational tool, assisting people in learning Braille in a fun and interactive way.
c. **Collaboration:** Partnering with educators and accessibility experts could bring valuable insights and ideas to improve the project's impact.

12 APPENDIX:

```
#include <Servo.h>
#include <SoftwareSerial.h>

// Define motor control constants
const int motorPins[6] = {3, 5, 6, 9, 10, 11};
Servo motors[6];

// Braille Alphabet Mapping (a-z)
bool brailleAlphabet[26][6] = {
  {1, 0, 0, 0, 0, 0}, // a
  {1, 1, 0, 0, 0, 0}, // b
  {1, 0, 0, 1, 0, 0}, // c
  {1, 0, 0, 1, 1, 0}, // d
  {1, 0, 0, 0, 1, 0}, // e
  {1, 1, 0, 1, 0, 0}, // f
  {1, 1, 0, 1, 1, 0}, // g
  {1, 1, 0, 0, 1, 0}, // h
  {0, 1, 0, 1, 0, 0}, // i
  {0, 1, 0, 1, 1, 0}, // j
  {1, 0, 1, 0, 0, 0}, // k
  {1, 1, 1, 0, 0, 0}, // l
  {1, 0, 1, 1, 0, 0}, // m
  {1, 0, 1, 1, 1, 0}, // n
  {1, 0, 1, 0, 1, 0}, // o
  {1, 1, 1, 1, 0, 0}, // p
  {1, 1, 1, 1, 1, 0}, // q
  {1, 1, 1, 0, 1, 0}, // r
  {0, 1, 1, 1, 0, 0}, // s
  {0, 1, 1, 1, 1, 0}, // t
  {1, 0, 1, 0, 0, 1}, // u
  {1, 1, 1, 0, 0, 1}, // v
  {0, 1, 0, 1, 1, 1}, // w
  {1, 0, 1, 1, 0, 1}, // x
  {1, 0, 1, 1, 1, 1}, // y
  {1, 0, 1, 0, 1, 1} // z
};

void setup() {
  Serial.begin(9600);
  Serial.println("Hello Arduino\n");
  for (int i = 0; i < 6; i++) {
    motors[i].attach(motorPins[i]);
  }
}
```

```
void loop() {
  if (Serial.available() > 0) {
    char input = Serial.read();
    if (isAlpha(input)) {
      bool* brailleDots = charToBraille(tolower(input));
      controlMotors(brailleDots);
     while (!Serial.available()) {
        // Wait here until next input is available
        delay(10); // Small delay to prevent locking up
      }
    }
  }
}

bool* charToBraille(char input) {
  return brailleAlphabet[input - 'a'];
}

void controlMotors(bool dots[6]) {
  for (int i = 0; i < 6; i++) {
    if (dots[i]) {
      motors[i].write(0);
    } else {
      motors[i].write(90);
    }
  }
}
```