SMART ASSISTIVE ATM SYSTEM FOR BLIND AND VISUALLY IMPAIRED USERS

PROJECT REPORT

Submitted By

KAVYA. A. J 311621106013

NEHA. L 311621106022

PRIYA. R 311621106026

In partial fulfillment for the award of degree

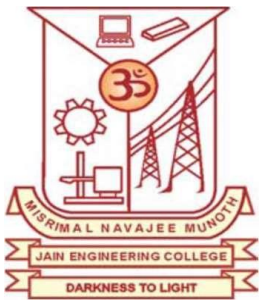
of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

MISRIMAL NAVAJEE MUNOTH JAIN ENGINEERING COLLEGE



ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

SMART ASSISTIVE ATM SYSTEM FOR BLIND AND VISUALLY IMPAIRED USERS

PROJECT REPORT

Submitted By

KAVYA. A. J 311621106013

NEHA. L 311621106022

PRIYA. R 311621106026

In partial fulfillment for the award of degree

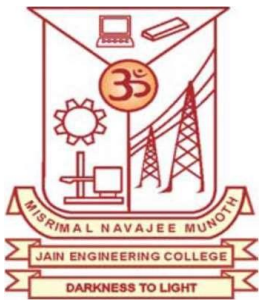
of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

MISRIMAL NAVAJEE MUNOTH JAIN ENGINEERING COLLEGE



ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"SMART ASSISTIVE ATM SYSTEM FOR BLIND AND VISUALLY IMPAIRED USERS"** is the Bonafide work of **KAVYA. A. J (311621106013), NEHA. L (311621106022), PRIYA. R (311621106026)** who carried out the project work under my supervision.

SIGNATURE

Dr. P. VENUGOPAL, M.E., Ph.D.,
HEAD OF THE DEPARTMENT

Department of Electronics and Communication Engineering,
Misrimal Navajee Munoth Jain Engineering College,
Thoraipakkam,
Chennai-600 097

SIGNATURE

MS. P. LAVANYA, B.E, M.E.,
SUPERVISOR
Associate Professor

Department of Electronics and Communication Engineering,
Misrimal Navajee Munoth Jain Engineering College,
Thoraipakkam,
Chennai-600 097

Submitted for the project work and viva voce Examination held on _____.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our gratitude and sincere thanks to our respected, Secretary **Dr. Harish L Metha (Admin),** Secretary **Shri. L. Jaswant Munoth (Academic)** and our beloved Principal **Dr. C. C. Christopher** for providing all kind of infrastructure for the successful completion of this project.

We are indebted to our Professor and Head, Department of Electronics and Communication Engineering, **Dr. P. Venugopal, M.E, Ph.D** and our project coordinator **Dr.K. Jayabharathi, M.E, Ph.D,** Associate Professor, for their enthusiastic motivation and continuous encouragement which inspired us a lot in completing this project.

We would like to express our gratitude to our internal guide **Ms. P. Lavanya, B.E, M.E.** Associate professor who has provided us his support, guidance and valuable suggestions that helped us to complete our project successfully.

Finally, we are grateful to our parents and friends for their constant encouragement and support to complete project successfully.

ABSTRACT

The Smart Assistive ATM for Blind and Visually Impaired Users is a pioneering solution designed to redefine accessibility in banking by providing a user-friendly, inclusive automated teller machine specifically tailored for individuals with visual impairments. This innovative system integrates OpenCV (Open Source Computer Vision) technology to enable features such as facial recognition, user detection, and adaptive interface responses, ensuring a secure and personalized experience for every user. By eliminating the need for traditional card-based access and complex menu navigation, the ATM allows blind users to independently carry out essential banking tasks such as cash withdrawal, balance inquiry, and PIN entry. The system is equipped with audio guidance that delivers real-time spoken instructions, while haptic feedback and a Braille-style interface provide tactile cues and input/output support for users who are proficient in Braille. An LCD with high-contrast visuals is included for those with partial vision, enhancing readability and usability. The entire operation is managed by an Arduino Mega microcontroller, which coordinates hardware components like keypads, audio modules, Braille displays, and security mechanisms.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

SYMBOL	DESCRIPTION
KB	Kilobytes
mA	Milliampere
MHz	Megahertz
V	Voltage
A	Ampere
Vin	Input Voltage
Vout	Output Voltage
I/O	Input/Output

LIST OF ABBREVIATION

ABBREVIATION	DESCRIPTION
ATM	Automated Teller Machine
LCD	Liquid Crystal Display
USB	Universal Serial Bus
LBPH	Local Binary Pattern Histogram
OCR	Optical Character Recognition
AI	Artificial Intelligence
CV	Computer Vision
ML	Machine Learning
CNN	Convolutional Neural Network
GPIO	General Purpose Input Output
DC	Direct Current
I2C	Inter-Integrated Circuit
EEPROM	Electrically Erasable Programmable ROM

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The Smart Assistive ATM System is a specialized prototype designed to enhance banking accessibility for blind and visually impaired users. At the heart of the system is the Arduino Mega microcontroller, which acts as the central control unit, seamlessly managing interactions among all integrated components. This setup ensures smooth operation and a user-friendly experience tailored to individuals with visual impairments.

For secure user authentication, the system incorporates OpenCV-based face detection technology. This eliminates the need for PIN entry, offering a contactless and efficient method to verify the account holder's identity. By leveraging image processing techniques, the system ensures that access is granted only to the authorized user, enhancing overall security.

To enable interaction, a tactile push-button interface functions as a Braille-style keypad, allowing users to input the required transaction amount through touch. A servo motor-based Braille display then provides real-time feedback, dynamically changing to reflect the options selected, thereby supporting accurate and independent usage.

Further improving accessibility, a DF Player module is used to deliver clear voice prompts, guiding the user through each transaction step. Additionally, an LCD screen displays transaction status and feedback for partially sighted users or supervisors. This comprehensive combination of technologies delivers an inclusive, safe, and efficient ATM experience for users with visual challenges.

1.2 OBJECTIVE

The main objective of this project is to design and develop an assistive ATM system that empowers blind and visually impaired individuals to carry out banking transactions independently and securely. Traditional ATM interfaces rely heavily on visual cues, which pose significant challenges for users with visual impairments. This project aims to bridge that gap by offering an inclusive and user-friendly solution.

A key focus is to incorporate face recognition using OpenCV to replace conventional PIN-based authentication, ensuring both security and ease of access. This contactless verification method eliminates the need for manual input of sensitive information, thus preventing unauthorized access and enhancing user convenience.

To support user interaction, the project aims to implement a tactile Braille input system and a servo-based Braille output mechanism. These components allow users to input the transaction amount and receive feedback in a format they can easily understand, promoting a seamless and confident user experience.

Another vital objective is to integrate voice guidance using a DF Player module to assist users throughout the transaction process. Combined with a visual LCD display for partially sighted users or administrators, the system is designed to be both inclusive and secure. Overall, the project strives to create a smart, accessible, and efficient ATM interface tailored specifically for the visually impaired community.

1.3 SCOPE OF THE PROJECT

This project presents an innovative, inclusive banking system designed to enhance accessibility for blind and visually impaired users while maintaining high security through advanced technologies.

At its core, the Arduino Mega microcontroller acts as the central hub, seamlessly coordinating all hardware components. Utilizing OpenCV for real-time face detection, the system ensures secure user verification by accurately identifying the account holder. To accommodate blind users, a tactile push button interface mimicking a Braille keypad allows easy input of transaction amounts.

A servo motor-driven Braille feedback mechanism offers intuitive, dynamic tactile responses, enhancing the user's interaction. Complementing this, the DF Player delivers clear voice instructions and prompts, ensuring blind users receive continuous audio guidance throughout the transaction process. Additionally, an LCD display provides visual transaction updates and status for enhanced transparency. This system creates a personalized, accessible, and secure banking experience for all users, particularly benefiting those with visual impairments, and represents a significant step forward in inclusive technology integration.

1.4 EXISTING SYSTEM

In the current banking infrastructure, standard ATMs are primarily designed for sighted users, relying on visual interfaces like screens, PIN pads, and on-screen menus. These machines lack accessibility features that cater to the needs of blind and visually impaired users, making it difficult or impossible for them to perform transactions independently.

Some existing ATMs attempt to improve accessibility by incorporating audio jacks for voice guidance, allowing users to plug in headphones and receive audio instructions. However, these features are often limited, inconsistent across banks,

and not widely available. Additionally, these systems still require the user to locate and press the correct buttons on a physical keypad, which can be difficult without tactile aids.

Security remains a major concern in the existing system. PIN-based authentication is vulnerable when blind users need assistance entering their PINs, compromising privacy and increasing the risk of fraud. These limitations discourage independent usage and can erode user confidence in ATM technology.

Overall, the existing ATM systems do not fully address the challenges faced by visually impaired users. They often lack integrated solutions that combine tactile input, voice output, and secure authentication, making them unsuitable for users who require a fully accessible and independent banking experience.

1.4.1 LIMITATION OF EXISTING SYSTEM

- **Lack of Accessibility:** Most ATMs are not equipped with tactile keypads or Braille, making it difficult for blind users to navigate the interface independently.
- **Limited Voice Assistance:** Only a few ATMs provide audio guidance, and even then, the functionality is often limited, requiring users to carry headphones and rely on unclear or poorly timed instructions.
- **Security Issues:** Blind users may need help from others to enter their PIN, compromising privacy and increasing the risk of fraud or unauthorized access.
- **Inconsistent Availability:** Accessible ATM features vary widely between banks and locations, making it unreliable for visually impaired users to find and use ATMs that meet their needs.
- **No Real-Time Tactile Feedback:** Traditional ATMs lack dynamic Braille or tactile feedback mechanisms, which are essential for blind users to confirm selections during transactions.

1.5 PROPOSED SYSTEM

The proposed system introduces a Smart Assistive ATM designed specifically to support blind and visually impaired users by combining multiple assistive technologies. At the core of the system is the Arduino Mega, which coordinates the operation of all components to deliver a seamless and user-friendly experience. The system ensures accessibility, security, and independence for users who are often excluded by traditional ATM designs.

To replace PIN-based authentication, the system uses OpenCV for face recognition, offering a secure, contactless method to identify and authenticate users. This eliminates the need for visually impaired individuals to input sensitive information through keypads, greatly reducing the risk of fraud or unauthorized access.

For transaction input, the system features a tactile push-button interface that mimics a Braille keypad, allowing users to enter the desired amount confidently. Additionally, a servo motor-driven Braille display dynamically outputs feedback in Braille format, guiding users through transaction options in a tactile manner, which greatly enhances usability and confidence.

1.5.1 PROPOSED SYSTEM ADVANTAGES

- Vision-Assisted Braille ATM System: Offers high-level security through face detection. Ensures complete accessibility for blind users via Braille input and feedback. Audio prompts enhance user independence, while Arduino-based control makes it affordable, customizable, and easy to integrate into existing banking infrastructure.
- AI-Based Inclusive Banking Terminal: Delivers a seamless and inclusive experience for visually impaired users. Face recognition adds an extra layer of security. Voice assistance and tactile feedback make banking more

intuitive, while Arduino Mega provides reliable multi-component control in real time

- Accessible Voice-Guided ATM with Braille Feedback: Improves ATM usability for blind users through a combination of tactile and auditory interfaces. Face verification boosts security. LCD enhances overall communication, and the system promotes independent access to banking services without requiring external assistance.
- The system also includes a DF Player audio module, which provides real-time, clear voice prompts that guide users through each step of the ATM process. For users with partial vision, a high-resolution LCD screen displays transaction status and progress. Together, these features create a secure, accessible, and empowering ATM experience for visually impaired users.

1.6 FLOW DIAGRAM

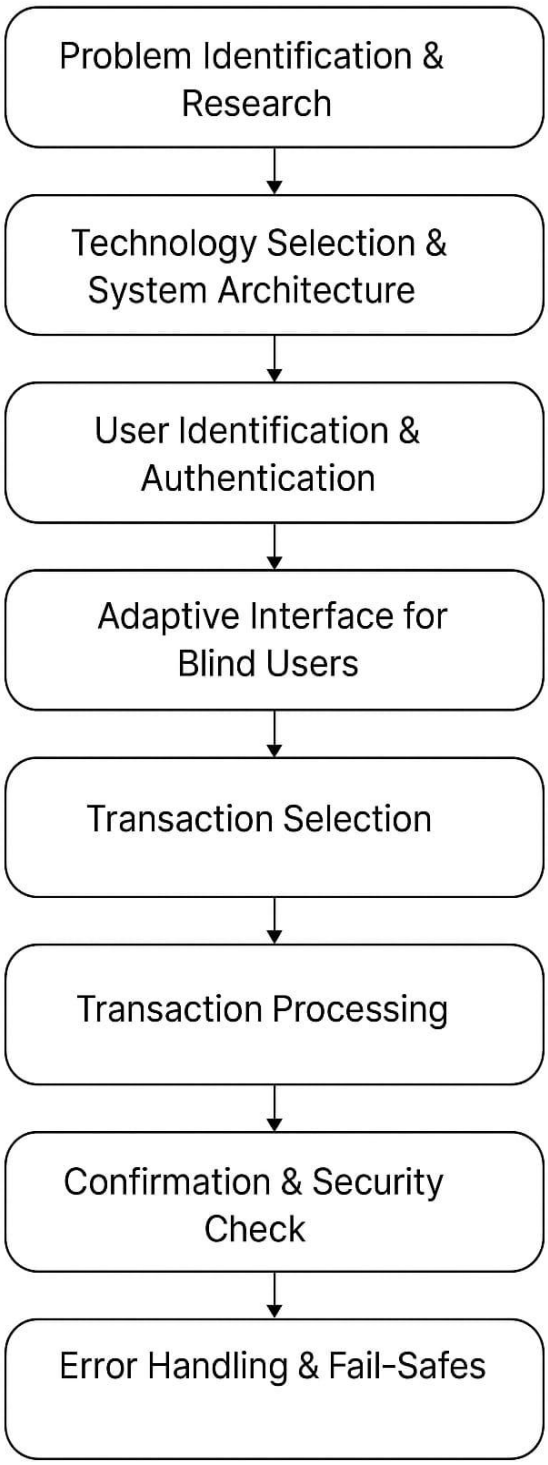


Figure 2.1 Flow chart

The development process of the "Smart Assistive ATM for Blind and Visually Impaired Users" follows a structured flow starting with problem identification and research to understand the challenges faced by visually impaired individuals in accessing ATM services independently and securely. This is followed by technology selection and system architecture design, where components such as OpenCV for facial recognition, Arduino Mega for hardware control, audio output modules, and Braille keypads are integrated to ensure a cohesive assistive system. In the user identification and authentication phase, secure and inclusive methods such as facial recognition, voice prompts, and optional PIN or Braille-based input are employed to validate users. The system then offers an adaptive interface tailored for blind users, providing intuitive voice guidance, tactile inputs, and minimal visual dependency. Users can then proceed to transaction selection, such as balance inquiry, withdrawal, or mini-statement, through audio-assisted prompts. Once selected, the transaction processing unit securely handles the request, maintaining data integrity and interaction with the banking system. A confirmation and security check is implemented to ensure that the transaction details are verified before execution. Additionally, the system is equipped with error handling mechanisms to detect issues like failed recognition, incorrect inputs, or hardware faults, while fail-safes such as emergency cancel operations and audible alerts ensure user safety. The overall system is designed to enhance accessibility, independence, and confidence for blind users in managing their banking needs.

1.7 ORGANIZATION OF THE REPORT

The project report is organized as follows : In Chapter 2, The literature survey is discussed . In Chapter 3, the proposed system and system specification are discussed. In Chapter 4, the software requirements of the project are discussed. In Chapter 5, the system implementation, output and the result are discussed. In Chapter 6, finally the conclusion is drawn and some future work also suggested.

CHAPTER 2

LITERATURE SURVEY

Christiawan et al. [1] introduced the Finger Shield ATM system to combat ATM skimming in Indonesia. Presented in October 2018 at ISESD, the system combines fingerprint biometrics with smart card technology for dual-layer authentication. This method significantly strengthens ATM security by ensuring both biometric and card verification. It is particularly effective in minimizing unauthorized access and fraudulent withdrawals. The system also promotes user accountability by requiring physical presence for authentication.

Divyans Mahansari and Uttam Kumar Roy [2] proposed an NFC-based ATM authentication system. The idea, shared at ICCST in October 2019, enables cardless transactions using smartphones with NFC. By swiping the device near an ATM reader, users authenticate without a physical card. This reduces risks like skimming, duplication, and hardware damage. The system is cost-effective and encourages faster transaction times.

Gokul et al. [3] developed a smart ATM system incorporating RFID and fingerprint biometrics. Their work was presented at ICSSIT in August 2020 and features real-time user alerts with location, time, and status. The system also includes vibration and flame sensors along with camera monitoring. These features enhance both physical safety and transaction-level security.

Joseph B. Awotunde et al. [4] addressed PIN security issues with a fingerprint-based ATM design. Introduced in 2014 in IJAIS, the prototype combines biometrics with AI-driven voice recognition. This enhances user access control and safety, especially for women. It also provides a reliable alternative to traditional PINs.

Kohbalan Moorthy et al. [5] developed a cloud-enabled ATM with fingerprint-based hybrid authentication. Presented in IJSECS in 2024, the system streamlines transaction security while supporting emergency responses. It improves user-friendliness and protects vulnerable users. The approach is particularly geared toward safe ATM access for women.

M. Mohamed Idhris et al. [6] addressed ATM cash flow uncertainty through a real-time replenishment model. Published in the March 2017 issue of IJETT, the study accounts for increased demand during weekends and holidays. The model optimizes refill timing to balance cost and availability. It aims to maintain user satisfaction while reducing idle cash.

Mohammadhossein Kiyaei [7] proposed a dynamic ATM cash replenishment method using AI. Introduced at CSICC in 2021, the system applies a Double Deep Q-Network (DQN) model to respond to fluctuating demands, such as those during COVID-19. This smart planning reduces operational costs and cash shortages. It ensures optimal ATM performance in smart city environments. The approach adapts to real-time usage patterns, making it scalable and responsive.

Paschal A. Ochang and Paulinus O. Ofem [8] proposed a biometric ATM access system to improve women's safety. Published in the 2017 issue of IJANA, the system integrates fingerprint verification with emergency alerts. This dual-layer design offers secure access and personal safety. It represents a meaningful step toward women-focused security solutions. The integration with emergency services makes it highly responsive to real-world threats.

Sayan Hazra [9] designed a smart ATM prototype using IoT and computer vision. The system, presented at DevIC in 2019, is based on a Raspberry Pi and supports facial recognition and surveillance. These features improve the ATM's interaction with users. It contributes to smarter, safer, and more responsive

banking services. The use of low-cost hardware makes it a scalable solution for rural areas as well.

Shivam Kumar Rajput et al. [10] introduced a dual-authentication ATM model using fingerprints and OTPs. Published in the 2019 issue of IJARIIT, the system sends a 4-digit OTP via GSM post-fingerprint verification. This two-step security enhances protection against fraud. It ensures secure and trustworthy ATM transactions. The inclusion of GSM ensures reach even in low-internet areas.

CHAPTER 3

SYSTEM DESIGN

3.1 PROPOSED SYSTEM DESIGN

The proposed smart ATM system is designed to enhance the accessibility of banking services for blind and visually impaired users, empowering them to independently perform transactions. The system is a combination of multiple modules working cohesively to provide a seamless user experience. At the heart of the design is the Arduino Mega, which controls the interaction between the various hardware peripherals like the servo motor, DF Player, LCD, and push buttons.

3.2BLOCK DIAGRAM

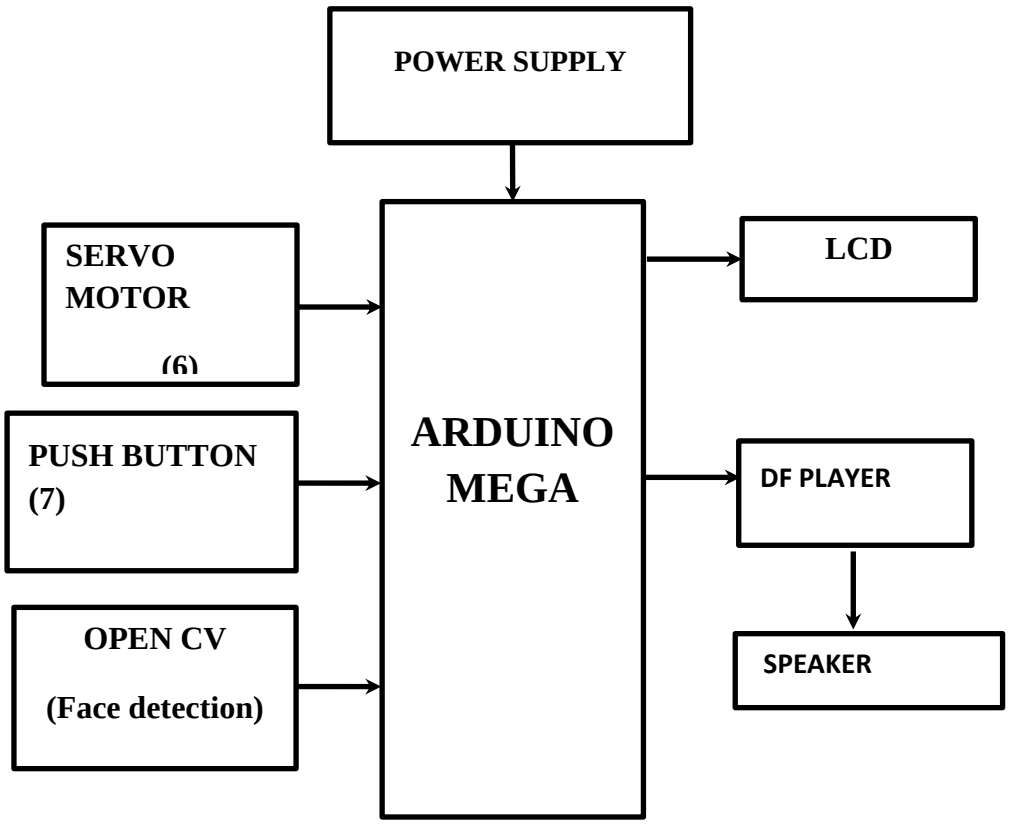


Figure 3.1 Block Diagram

The system utilizes haptic feedback via the servo motor, auditory feedback using the DF Player, and visual guidance through the LCD to guide users through the transaction process. One of the key features of this system is the integration of computer vision through OpenCV, which allows gesture recognition. These components together ensure that the ATM system can be fully operated without the need for sight, promoting financial inclusivity and privacy.

3.2.1 BLOCK DIAGRAM DESCRIPTION

Power Supply: Provides the necessary electrical power for the entire system.

Arduino Mega: Acts as the central controller, processing inputs and controlling outputs.

Push Button: Allows user input to trigger actions or events.

Servo Motor: Provides precise angular movement based on control signals.

DF Player: Plays audio files from a microSD card for audio feedback.

Speaker: Outputs sound or audio signals generated by the DF Player.

LCD: Displays text or data to the user for interaction or feedback.

3.3 INPUT MECHANISM

3.3.1 Push Button Inputs

Push buttons serve as the primary tactile input mechanism for the user in the smart ATM system. These buttons are large, easy to press, and designed specifically for users with visual impairments. Each button is associated with a specific function, such as "Confirm," "Cancel," "Enter," or "Clear." The tactile design allows the user to feel the button and confirm its location, making it simple to interact with the system even without sight. The buttons are positioned to be easily accessible and ergonomically placed within the ATM.

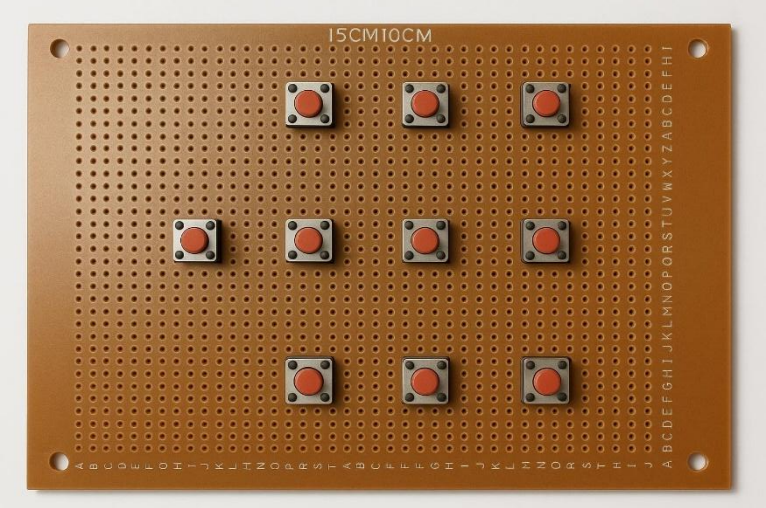


Figure 3.2 Push buttons for user input

The push buttons are connected directly to the Arduino Mega, which interprets each press and triggers corresponding actions within the system. Upon pressing a button, the system responds with auditory feedback through the DF Player and haptic feedback through the servo motor, ensuring that the user receives a clear and immediate confirmation. This combination of audio, tactile, and visual feedback ensures that users are never uncertain about their actions, making the system easy to navigate.

In addition to their tactile design, the push buttons are engineered for durability, ensuring they can withstand the frequent usage typically seen in public ATM

locations. The materials used for the buttons are resistant to wear and tear, which guarantees longevity. Their large size ensures that even individuals with limited dexterity can use them effectively. By offering a simple and intuitive interface, the push buttons allow the visually impaired user to interact with the ATM without requiring any specialized training.

3.4 FEEDBACK MECHANISM

3.4.1 Servo Motor-Based Haptic Feedback

In this assistive ATM system, the servo motor plays a crucial role in providing haptic feedback to the user. This feedback is essential for creating a multisensory experience, allowing the user to receive tactile confirmation of their actions.

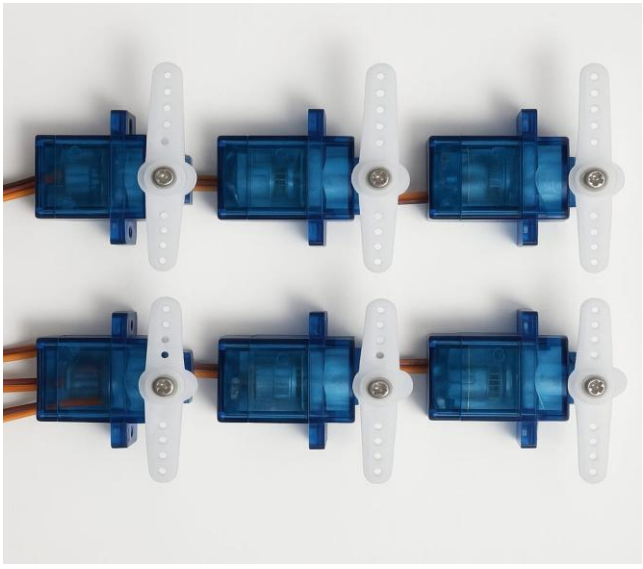


Figure 3.3 Braille servo grid

When a user interacts with the ATM by pressing a button or making a gesture, the servo motor activates and moves slightly, providing a physical response that confirms the system has received the input. This feedback is an important accessibility feature for individuals who cannot rely on visual cues alone.

The servo motor is programmed to run for two seconds when verifying the accuracy of a user's input. If the user presses a correct button or performs the

right gesture, the servo motor's to-and-fro motion acts as confirmation that the input was registered successfully.

To power the servo motor, a dedicated 12V power supply is used. This ensures that the motor operates smoothly without interfering with the other components of the system, such as the Arduino or LCD. By isolating the servo motor’s power supply, the design minimizes the risk of voltage fluctuations, ensuring stable operation and longevity of the motor. The servo motor's durable and responsive nature makes it ideal for this application, offering consistent and reliable feedback even with heavy usage.

3.5 USER INTERFACE AND MODULE

3.5.1 LCD Display Interface

Figure 3.4 LCD

While the primary user interface for visually impaired users is based on audio and haptic feedback, the LCD screen remains a critical component of the system. The LCD display provides essential visual information to partially sighted users, service personnel, or any others assisting with the ATM transaction process. It displays the steps involved in the transaction, such as the selection of the transaction type, confirmation of inputs, and error messages in case of invalid

actions. The display provides clear and simple text-based instructions, ensuring that all users can follow the system's progress.

The LCD interface is powered by the Arduino Mega, which controls and updates the screen based on user inputs and system events. The screen offers high visibility and clarity, with large, legible fonts for users with partial sight. The integration of the LCD alongside the audio and haptic systems ensures a multifaceted approach to user guidance, accommodating various levels of vision impairment and ensuring that all users receive the necessary feedback to navigate the ATM. The information displayed on the LCD complements the auditory instructions by showing textual confirmations or alerts at each stage of the transaction. For example, when a user selects an option, the LCD confirms the choice with a clear message, such as "Option Selected: Withdraw Funds." This helps users verify that their inputs are being processed correctly. It also serves as an error indicator, displaying helpful error messages if something goes wrong during the transaction, thus making the system more transparent and user-friendly.

3.6 AUDIO OUTPUT SYSTEM

3.6.1 DF Player and Speaker Module

The DF Player Mini is an essential component of the audio output system, playing pre-recorded audio files to guide the user through each step of the ATM transaction. The module can store multiple audio files on a microSD card, and it is controlled by the Arduino Mega through serial communication. The DF Player Mini allows for real-time audio feedback, which is crucial for visually impaired users, providing them with audible cues such as "Select your transaction type,"

and “Transaction complete.” The audio output is relayed through a speaker connected to the DF Player. The speaker ensures that the audio is loud enough to be heard clearly in an indoor environment, such as an ATM kiosk.

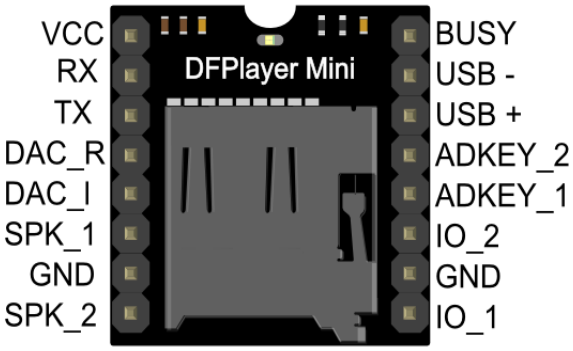


Figure 3.5 DF Player

The sound quality is optimized for clarity and ease of understanding, ensuring that users can follow instructions without confusion. The system also includes volume control, enabling the user to adjust the volume to suit the environment, ensuring that it is neither too loud nor too soft.

Overall, the DF Player and speaker system enhances the accessibility of the ATM, ensuring that visually impaired users can fully engage with the system. It offers an intuitive and efficient way for users to perform transactions independently, with immediate auditory feedback that allows them to make informed decisions and complete their tasks confidently.

3.7 MICRO CONTROLLER

3.7.1 Arduino Mega

The Arduino Mega serves as the core controller in the Smart Assistive ATM System. It is responsible for managing and coordinating the system’s interactions

between various hardware components such as the servo motor, LCD display, DF Player, and push buttons. The Arduino Mega was chosen due to its increased number of input/output pins, which is crucial for handling multiple peripherals simultaneously. It has 54 digital I/O pins, 16 analog inputs, and 4 serial communication ports, providing ample connectivity options for the various components of the system.

Arduino Mega's versatility in handling multiple tasks concurrently is vital for the real-time operations of the ATM.The Arduino Mega is programmed in Embedded C using the Arduino IDE, and it communicates directly with all the system components, ensuring the ATM system operates without delays. The board runs in real-time, ensuring precise and immediate feedback to users' actions.

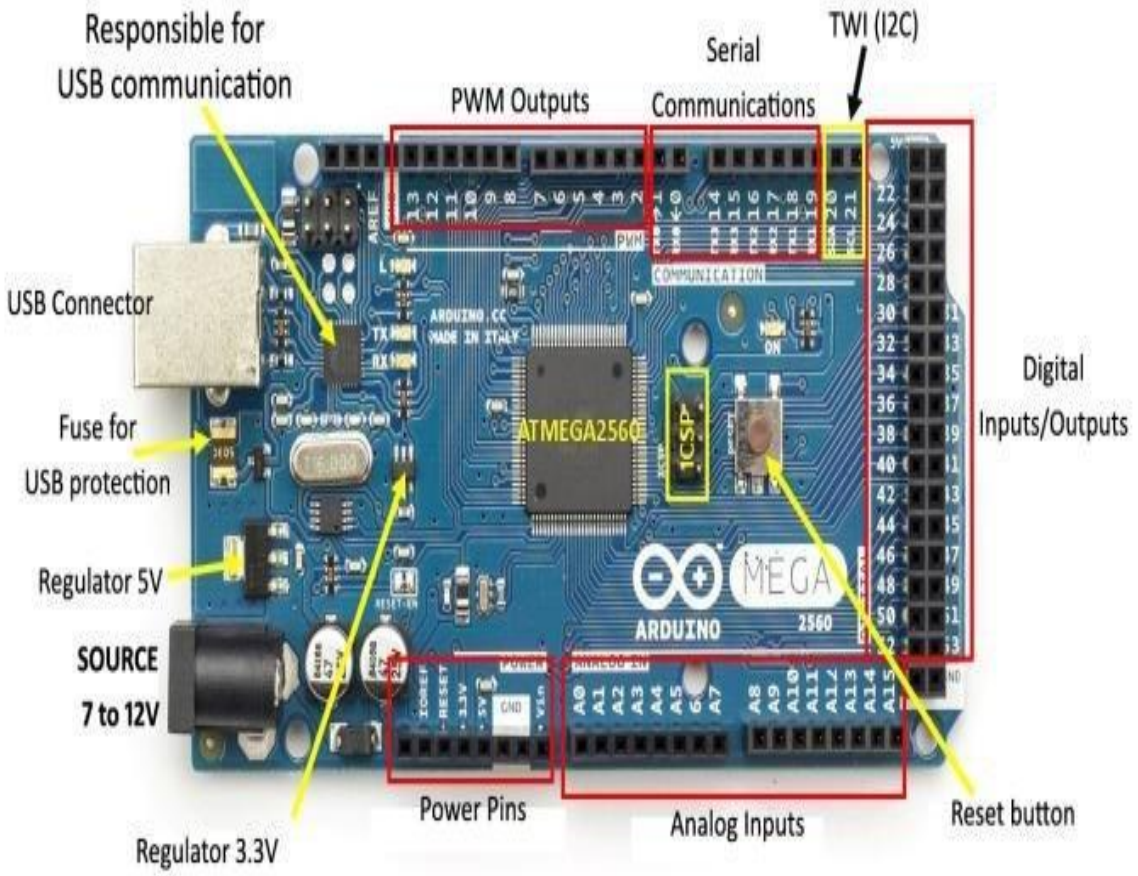


Figure 3.6 ATmega2560

The Arduino Mega is powered using a 12V DC supply, ensuring stable performance for the entire system. Separate power supplies are used for the servo motor (which requires more power) and the rest of the components to prevent power fluctuations. This ensures that high-demand components like the servo motor do not interfere with the performance of the microcontroller or other peripherals.

3.7.2 TECHNICAL SPECIFICATIONS

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by boot loader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length, Width	101.52 mm,53.3mm

CHAPTER 4

SOFTWARE REQUIREMENTS

4.1 SOFTWARE DESCRIPTION

The software implementation of the smart ATM system utilizes a combination of embedded systems programming and high-level scripting for computer vision tasks. The embedded side of the project is written in Embedded C and implemented using the Arduino IDE. This code controls the hardware components, such as the servo motor, LCD display, and push buttons, ensuring seamless interaction between the user and the system. Embedded C was chosen for its low-level access to hardware, which ensures precise control over the peripherals connected to the Arduino Mega.

On the other side, the computer vision module is implemented using Python and OpenCV, enabling gesture recognition and interaction. Python was chosen because of its ease of use, powerful libraries, and compatibility with OpenCV. The Python scripts run on the host PC, which connects to the ATM system to process camera input. OpenCV's algorithms allow for real-time image processing and analysis, interpreting the user's gestures and translating them into commands for the Arduino. This setup makes the system more dynamic and user-friendly, enabling users to interact with the ATM using natural hand movements.

4.1.1 PROGRAMMING LANGUAGES

The software for this project is developed using a combination of Embedded C and Python. Embedded C is utilized for programming the Arduino Mega, which is the central microcontroller of the system. The use of Embedded C allows for precise control over hardware components such as the servo motor, LCD screen, and push buttons. The Arduino IDE serves as the development platform for compiling and uploading the code to the Arduino Mega. This language was

chosen due to its efficiency in low-level hardware control, allowing the system to interact seamlessly with physical components.

Python, in conjunction with OpenCV, is employed for the computer vision part of the project. Python was selected for its simplicity and readability, making it easier to implement complex image processing algorithms and handle real-time interaction. OpenCV, a powerful computer vision library, enables the ATM system to recognize hand gestures and interpret visual inputs from the user. Python's ability to integrate well with OpenCV allows for real-time gesture recognition, making it a key component of the system's accessibility features.

4.1.2 REQUIRED LIBRARIES AND TOOLS

The successful implementation of the smart ATM system relies on a set of libraries and tools that support the development and integration of various components. For the embedded programming part, the Arduino IDE is used for writing and uploading code to the Arduino Mega. This IDE supports the necessary libraries for controlling the servo motor, interfacing with the LCD, and handling input from the push buttons. The Arduino core libraries, along with additional libraries like Servo.h, LiquidCrystal.h, and DFPlayerMini_Fast.h, are essential for controlling the hardware peripherals and enabling smooth system operation.

On the computer vision side, OpenCV is the core library for processing visual input and implementing gesture recognition. OpenCV provides a set of functions for capturing images from the Pi Camera, detecting objects or gestures, and processing the visual data in real-time.

Anaconda Navigator is an integral part of the development setup, ensuring that the Python environment is properly configured with all the necessary dependencies. Lastly, the system uses the Arduino IDE for embedded C

programming, OpenCV and Python for computer vision tasks, and Anaconda Navigator for managing the Python environment.

4.2 ANACONDA NAVIGATOR

Anaconda Navigator is a comprehensive graphical user interface that simplifies the process of managing Python environments, libraries, and dependencies. In the context of this project, Anaconda Navigator plays a vital role in managing the Python environment and ensuring that all required libraries, including OpenCV, NumPy, and others, are correctly installed. Its user-friendly interface allows developers to easily create, manage, and switch between different environments, ensuring that the project's dependencies remain compatible and isolated from other Python projects.

With Anaconda, developers can quickly install, update, or remove libraries with a few clicks, reducing the complexity of managing software packages. This tool helps streamline the setup process for the computer vision module, ensuring that all the necessary tools are in place before development begins. It also provides an easy-to-use platform for managing the project's dependencies, which is especially helpful in avoiding version conflicts that could arise with multiple libraries.

4.3 OPENCV

OpenCV (Open Source Computer Vision Library) is the backbone of the computer vision aspect of this project. It is a powerful and widely-used library that provides a rich set of tools for real-time image processing. OpenCV is used in the smart ATM system to recognize hand gestures or other visual inputs, enabling a more interactive and intuitive user experience. This system replaces or supplements traditional methods like card swipes, allowing users to interact with the ATM through natural hand movements or gestures.

The main function of OpenCV in this project is to process the images captured by the camera. It analyzes the video feed in real time, identifying specific gestures or actions made by the user. For instance, OpenCV is used to detect when a user moves their hand to signal an action like entering a PIN or confirming a transaction. The processed information is then sent to the Arduino, which triggers the appropriate actions, such as providing audio feedback or controlling the servo motor for haptic feedback.

4.4 ARDUINO IDE

The Arduino Integrated Development Environment (IDE) is the primary tool used to develop and upload code to the Arduino Mega in this project. It provides a simple and user-friendly interface for writing Embedded C programs that control various hardware components like the servo motor, DF Player, push buttons, and LCD.

Additionally, the serial monitor built into the Arduino IDE is essential for real-time debugging. It allows developers to print status messages and values directly from the Arduino board, making it easier to verify whether sensor data is being read correctly or if specific conditions are being met during execution. Overall, the Arduino IDE is a central part of this project's development environment, bridging the gap between hardware and software to deliver a cohesive assistive technology solution.

CHAPTER 5

SYSTEM IMPLEMENTATION AND RESULT

5.1 CIRCUIT DIAGRAM

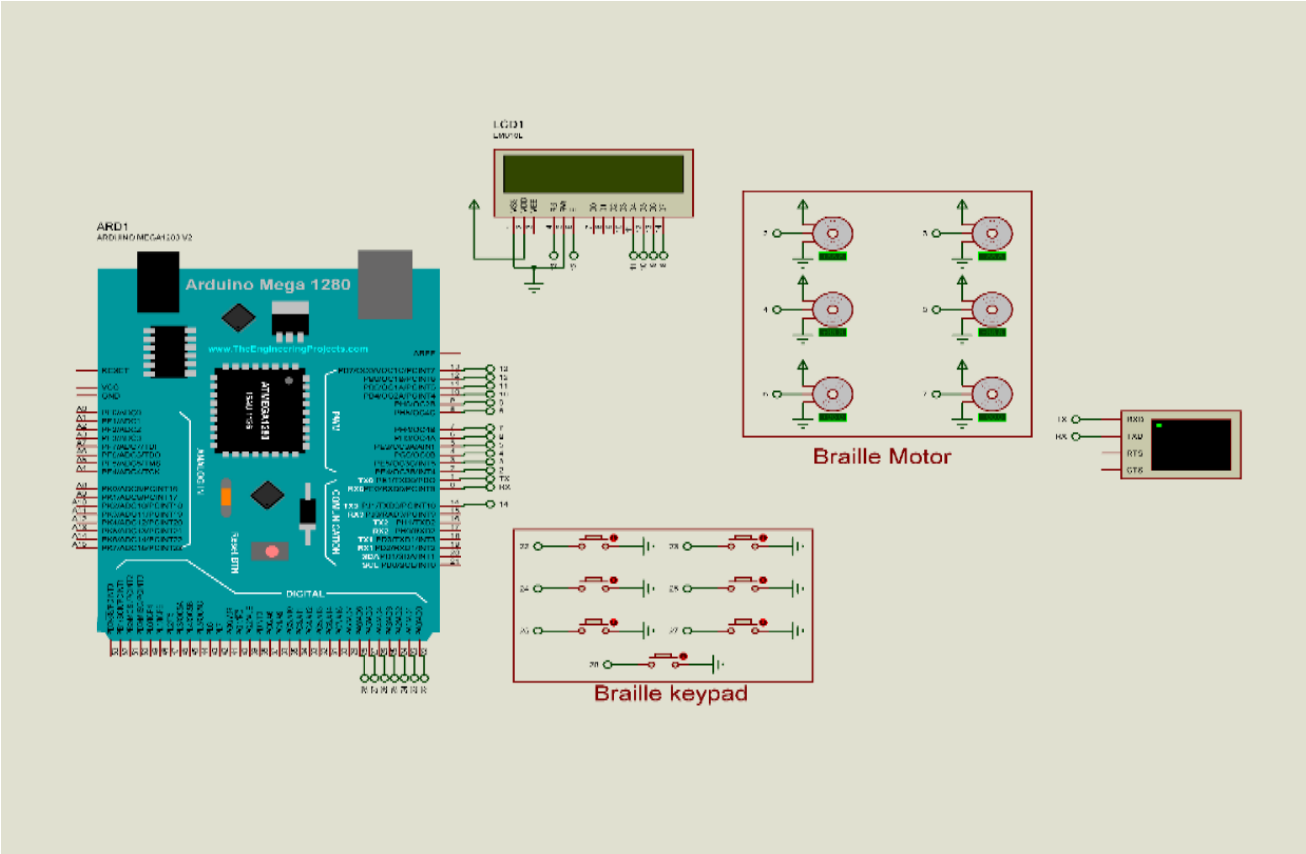


Figure 5.1 Circuit diagram

5.1.1 WORKING PROCESS

The working of the circuit begins when a visually impaired user interacts with the Braille keypad. Each switch on the keypad corresponds to one of the six dots used in Braille characters. When a combination of keys is pressed, the Arduino Mega 1280 reads the inputs through its digital input pins and interprets the intended character or command. This input is processed internally to determine the appropriate response or output, such as displaying a menu option, processing a transaction request, or navigating through the system.

Once a valid input is recognized, the system provides feedback in two ways. For tactile feedback, the Arduino controls six micro motors, each aligned with a Braille dot position. These motors raise or lower small pins to form the correct Braille pattern, allowing the user to read the output through touch. This mechanism enables users to feel responses such as balance details or menu options physically. For visual feedback—used mainly by developers or assisting personnel—a 16x2 LCD module displays the same information in text form.

To enhance security and interactivity, the device operates in a loop, consistently checking for new keypad input or external data. It maintains real-time responsiveness by processing input, updating outputs (LCD or Braille motor), and managing communication simultaneously. As a result, the system provides an inclusive ATM experience that is user-friendly, secure, and tailored to the needs of blind users, bridging accessibility gaps in banking.

5.2 SYSTEM PROTOTYPE

The prototype of the Smart Assistive ATM for Blind and Visually Impaired Users integrates an Arduino Mega 2560 as the central controller connected to a 16x2 LCD display, a custom Braille keypad made on a perfboard for tactile input, and eight servo motors configured to simulate Braille output by dynamically raising and lowering pins. A voice module with a speaker provides audio instructions and feedback, enabling hands-free interaction. The system includes motor driver modules (L298N) to control the servos, a USB-to-Serial converter for programming and debugging, and a regulated power supply to ensure stable operation. All components are interconnected using jumper wires, making the prototype a compact, accessible, and user-friendly simulation of a real ATM tailored for the blind and visually impaired community. The main objective of this prototype is to demonstrate an assistive ATM interface that enables blind and visually impaired users to perform banking transactions independently using

Braille-based input and output, voice guidance, and tactile feedback systems. The prototype focuses on enhancing accessibility, safety, and ease of use in financial services.

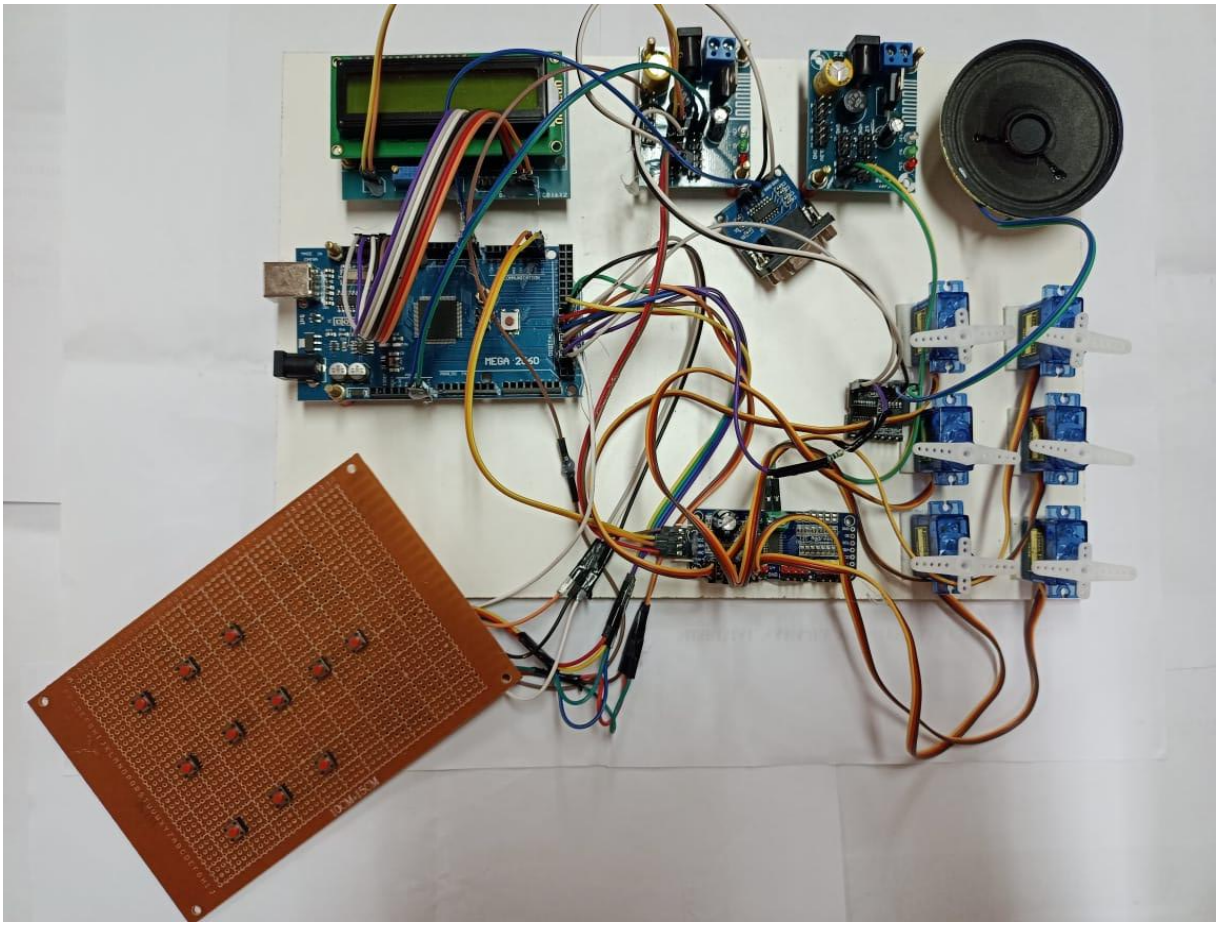


Figure 5.2 Outlook of the project hardware

5.3 OUTPUT

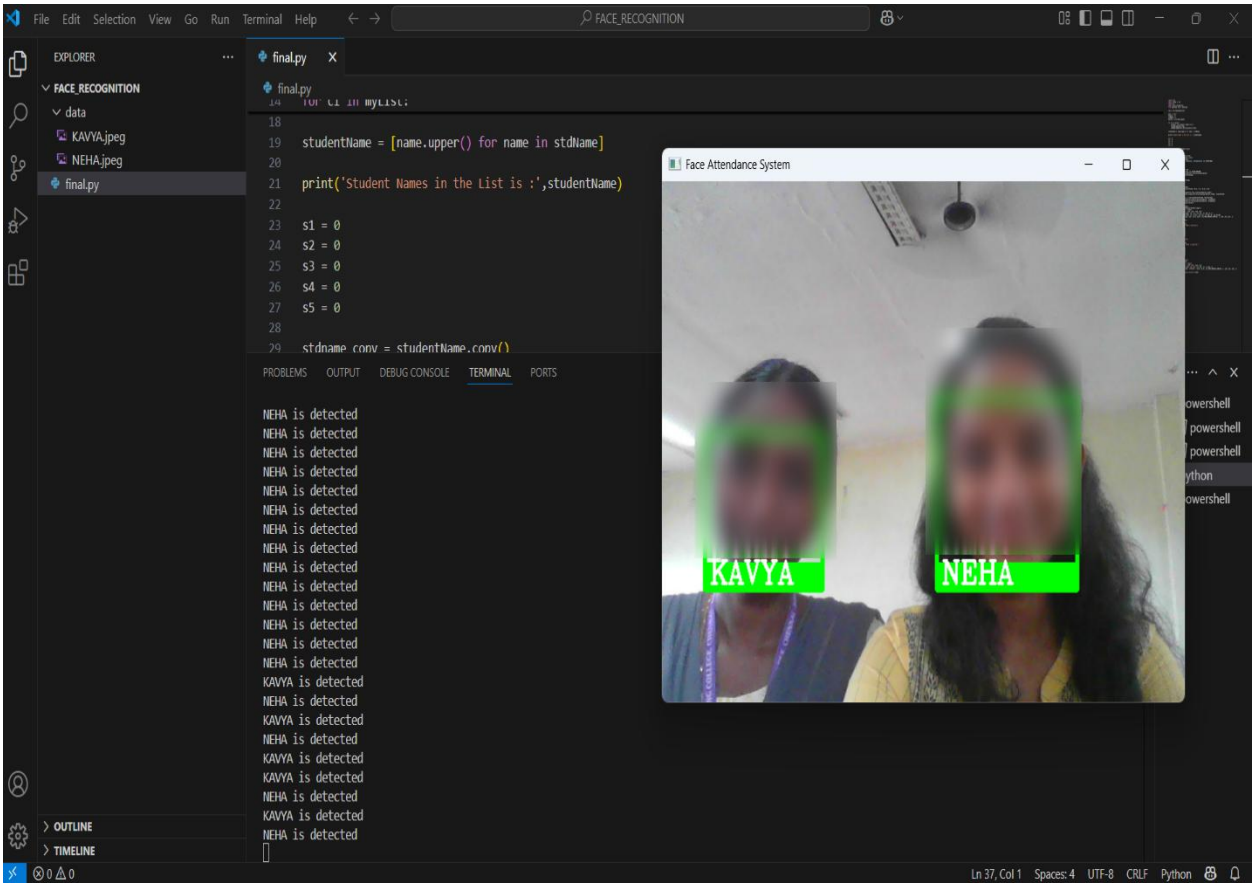


Figure 5.3 Face Recognition results

The face recognition module plays a vital role in the software implementation of the "Smart Assistive ATM for Blind and Visually Impaired Users." This module enhances security by enabling contactless user authentication through facial recognition, eliminating the need for PIN entry or card swiping, which can be challenging for visually impaired users. Developed using Python with OpenCV and the face_recognition library, the system is designed to recognize authorized users in real-time by comparing webcam input with pre-stored reference images. The application processes live video frames, detects faces, and matches them against encoded facial data from the dataset. When a match is identified, the

user’s name is displayed on the video feed within a bounding box, and the terminal logs confirmation messages such as “NEHA is detected” or “KAVYA is detected.” This setup confirms the system's ability to authenticate users accurately.. Overall, the face recognition component contributes significantly to the ATM’s accessibility and security.

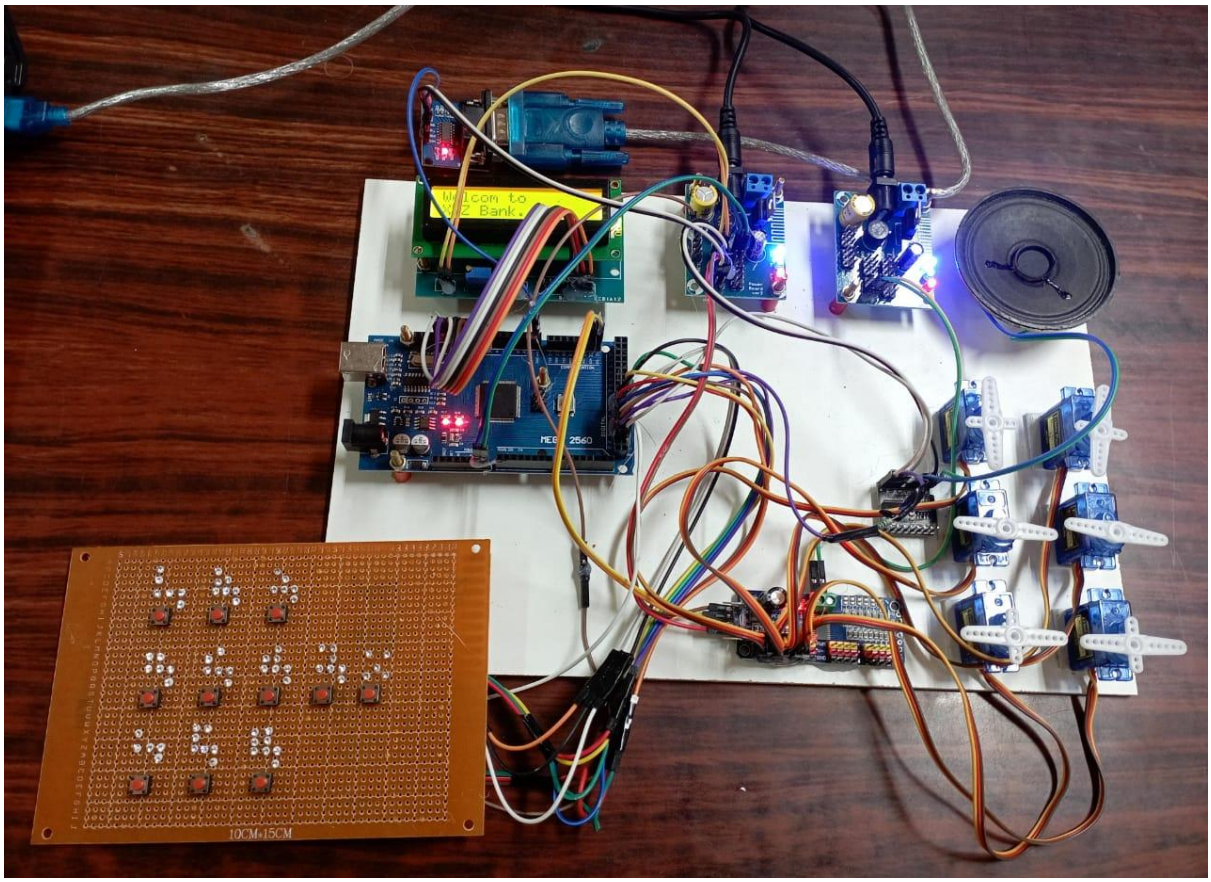


Figure 5.4 Hardware results

The hardware component of the "Smart Assistive ATM for Blind and Visually Impaired Users" is designed to facilitate accessible and secure ATM interactions through a combination of microcontroller control, tactile feedback, and audio guidance. The system is built around an Arduino Mega 2560 microcontroller, which serves as the central control unit interfacing with multiple peripheral

modules. A 4x3 tactile Braille-style keypad is provided for PIN input and menu navigation, enabling visually impaired users to interact with the system using touch. An LCD display is included to display transaction-related information, primarily for monitoring and debugging purposes. For audio feedback, a speaker is connected to provide voice guidance and read out instructions, enhancing usability for blind users. The system also integrates several servo motors to simulate ATM cash dispensing and Braille output mechanisms, offering a more realistic ATM experience. Additional modules such as voltage regulators, drivers, and communication interfaces ensure reliable operation. All components are mounted on a project baseboard for clarity and convenience, with clear wiring connections and modular organization. This hardware configuration supports the goal of creating a functional, assistive ATM prototype that bridges accessibility gaps in financial transactions.

CHAPTER 6

CONCLUTION & FUTURE ENHANCEMENT

6.1 CONCLUSION

The Smart Assistive ATM System has been successfully designed and implemented to provide a reliable, accessible, and user-friendly interface for blind and visually impaired individuals. Traditional ATMs pose challenges for users with visual impairments due to their reliance on visual cues. This project addresses those challenges by integrating tactile input, haptic feedback, audio output, and computer vision technologies into one cohesive and adaptive solution.

The use of Arduino Mega as the central microcontroller ensures real-time coordination between all hardware components such as the servo motor, LCD, DF Player, push buttons, and speaker. By incorporating assistive technologies such as haptic feedback for tactile confirmation and voice guidance for audio navigation, the system reduces dependency on third parties. This not only improves user confidence but also reinforces privacy and dignity during ATM use. Moreover, the modular and scalable design ensures that the system can be maintained and upgraded as needed, making it a future-proof solution.

In conclusion, the project fulfills its objective of providing a smart, inclusive ATM solution tailored for the visually impaired. It demonstrates how embedded systems, computer vision, and thoughtful hardware integration can be leveraged to solve real-world problems and promote digital equality. The success of this project sets the stage for further innovations and larger-scale implementation in public banking infrastructure.

6.2 FUTURE ENHANCEMENT

While the current system provides essential accessibility features, several improvements can further enhance functionality and user experience:

Biometric Authentication: Integration of fingerprint or iris scanners can provide an extra layer of security and eliminate the need for PIN entry altogether.

Voice Command Integration: Implementing voice recognition can allow users to interact with the ATM through spoken commands, reducing reliance on physical buttons.

Multi-language Support: Adding support for regional languages in audio output would make the system more user-friendly and inclusive for a diverse population.

Wireless Connectivity and IoT: Enabling cloud connectivity would allow remote monitoring, software updates, and real-time error reporting to maintenance teams.

APPENDICES

Module 1 To build a model for Face Recognition

```
import cv2
import numpy as np
import os
import face_recognition
from datetime import datetime
import serial
ser = serial.Serial('COM5',9600)
cap = cv2.VideoCapture(0)
path = 'data'
stdImg = []
stdName = []
myList = os.listdir(path)
for cl in myList:
    curimg = cv2.imread(f'{path}/{cl}')
    stdImg.append(curimg)
    stdName.append(os.path.splitext(cl)[0])
studentName = [name.upper() for name in stdName]
print('Student Names in the List is :',studentName)
s1 = 0
s2 = 0
s3 = 0
s4 = 0
```

```
s5 = 0
stdname_copy = studentName.copy()
def resize(img, size) :
    width = int(img.shape[1]*size)
    height = int(img.shape[0] * size)
    dimension = (width, height)
    return cv2.resize(img, dimension, interpolation= cv2.INTER_AREA)
def findEncoding(images)
    imgEncodings = []
    for img in images :
        img = resize(img, 0.50)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encodeimg = face_recognition.face_encodings(img)[0]
        imgEncodings.append(encodeimg)
    return imgEncodings
EncodeList = findEncoding(stdImg)
while True :
    success, frame = cap.read()
    Smaller_frames = cv2.resize(frame, (0,0), None, 0.25, 0.25)
    facesInFrame = face_recognition.face_locations(Smaller_frames)

   encodeFacesInFrame = face_recognition.face_encodings(Smaller_frames,
facesInFrame)
    for encodeFace, faceloc in zip(encodeFacesInFrame, facesInFrame) :
        matches = face_recognition.compare_faces(EncodeList, encodeFace)
```

```
        facedis = face_recognition.face_distance(EncodeList, encodeFace)
        matchIndex = np.argmin(facedis)
        if matches[matchIndex] :
            name = studentName[matchIndex].upper()
            y1, x2, y2, x1 = faceloc
            y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 3)
            cv2.rectangle(frame, (x1, y2-25), (x2, y2), (0, 255, 0),    cv2.FILLED)
            cv2.putText(frame, name, (x1+6, y2-6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
            if name == 'NEHA':
                s1 += 1
                if s1 >= 10:
                    print('NEHA is detected')
                    ser.write(b'A')
                    s1 = 0
                    s2 = 0
                    s3 = 0
                    s4 = 0
                    s5 = 0
            if name == 'KAVYA':
                s2 += 1
                if s2 >= 10:
                    print('KAVYA is detected')
                    ser.write(b'B')
```

```
                s1 = 0
                s2 = 0
                s3 = 0
                s4 = 0
                s5 = 0
        else:
            intruder_images=[]
            intruder_count=0
            y1, x2, y2, x1 = faceloc
            y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 3)
            cv2.putText(frame, 'INTRUDER', (x1+6, y2-6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
    cv2.imshow('Face Attendance System',frame)
    k = cv2.waitKey(1)
    if k == 27:
        break
cap.release()
```

EMBEDDED CODE:

BUTTON:

```
#ifndef BUTTON_H
#define BUTTON_H
#include "global.h"
char buttonToNumber(int btn1, int btn2, int btn3, int btn4, int btn5, int btn6, int
btn7, int btn8, int btn9, int btn10) {
  if (!btn1) return '1';
```

```
  if (!btn2) return '2';
  if (!btn3) return '3';
  if (!btn4) return '4';
  if (!btn5) return '5';
  if (!btn6) return '6';
  if (!btn7) return '7';
  if (!btn8) return '8';
  if (!btn9) return '9';
  if (!btn10) return '0';
  return 'x';
}
#endif
```

SERVO MOTOR:

```
#ifndef SERVO_H
#define SERVO_H
#include "global.h"
int angleToPulse(int ang) {
  int pulse = map(ang, 0, 180, SERVOMIN, SERVOMAX);
  Serial.print("Angle: "); Serial.print(ang);
  Serial.print(" pulse: "); Serial.println(pulse);
  return pulse;
}
void convertToBraille(char input) {
  int servosToRun[6] = { -1, -1, -1, -1, -1, -1 }; // Initialize with invalid indices
```

```
switch (input) {
  case 'A':
  case 'a':
  case '1':
    servosToRun[0] = 0;
    break;
  case 'B':
  case 'b':
  case '2':
    servosToRun[0] = 0; servosToRun[1] = 2;
    break;
  case 'C':
  case 'c':
  case '3':
    servosToRun[0] = 0; servosToRun[1] = 1;
    break;
  case 'D':
  case 'd':
  case '4':
    servosToRun[0] = 0; servosToRun[1] = 1; servosToRun[2] = 3;
    break;
  case 'E':
  case 'e':
  case '5':
    servosToRun[0] = 0; servosToRun[1] = 3;
```

```
      break;
    case 'F':
    case 'f':
    case '6':
      servosToRun[0] = 0; servosToRun[1] = 1; servosToRun[2] = 2;
      break;
    case 'G':
    case 'g':
    case '7':
      servosToRun[0] = 0; servosToRun[1] = 1; servosToRun[2] = 2;
servosToRun[3] = 3;
      break;
    case 'H':
    case 'h':
    case '8':
      servosToRun[0] = 0; servosToRun[1] = 2; servosToRun[2] = 3;
      break;
    case 'I':
    case 'i':
    case '9':
      servosToRun[0] = 1; servosToRun[1] = 2;
      break;
    case 'J':
    case 'j':
    case '0':
```

```
      servosToRun[0] = 1; servosToRun[1] = 2; servosToRun[2] = 3;
      break;
    case 'K':
    case 'k':
      servosToRun[0] = 0; servosToRun[1] = 4;
      break;
    case 'L':
    case 'l':
      servosToRun[0] = 0; servosToRun[2] = 2; servosToRun[4] = 4;
      break;
    case 'M':
    case 'm':
      servosToRun[0] = 0; servosToRun[1] = 1; servosToRun[4] = 4;
      break;
    case 'N':
    case 'n':
      servosToRun[0] = 0; servosToRun[1] = 1; servosToRun[3] = 3;
servosToRun[4] = 4;
      break;
    case 'O':
    case 'o':
      servosToRun[0] = 0; servosToRun[3] = 3; servosToRun[4] = 4;
      break;
    case 'P':
    case 'p':
```

```
      servosToRun[0] = 0; servosToRun[1] = 1; servosToRun[2] = 2;
servosToRun[4] = 4;

      break;

    case 'Q':

    case 'q':

      servosToRun[0] = 0; servosToRun[1] = 1; servosToRun[2] = 2;
servosToRun[3] = 3; servosToRun[4] = 4;

      break;

    case 'R':

    case 'r':

      servosToRun[0] = 0; servosToRun[2] = 2; servosToRun[3] = 3;
servosToRun[4] = 4;

      break;

    case 'S':

    case 's':

      servosToRun[1] = 1; servosToRun[2] = 2; servosToRun[4] = 4;

      break;

    case 'T':

    case 't':

      servosToRun[1] = 1; servosToRun[2] = 2; servosToRun[3] = 3;
servosToRun[4] = 4;

      break;

    case 'U':

    case 'u':

      servosToRun[0] = 0; servosToRun[1] = 1; servosToRun[4] = 4;
servosToRun[5] = 5;

      break;
```

```
    case 'V':
    case 'v':
      servosToRun[0] = 0; servosToRun[4] = 4; servosToRun[5] = 5;
      break;
    case 'W':
    case 'w':
      servosToRun[1] = 1; servosToRun[2] = 2; servosToRun[3] = 3;
servosToRun[5] = 5;
      break;
    case 'X':
    case 'x':
      servosToRun[0] = 0; servosToRun[3] = 3; servosToRun[4] = 4;
servosToRun[3] = 5;
      break;
    case 'Y':
    case 'y':
      servosToRun[0] = 0; servosToRun[1] = 1; servosToRun[2] = 3;
servosToRun[3] = 4; servosToRun[4] = 5;
      break;
    case 'Z':
    case 'z':
      servosToRun[0] = 0; servosToRun[3] = 3; servosToRun[4] = 4;
servosToRun[5] = 5;
      break;
    default: return;  }
  // Move servos from 0° → 90°
```

```
  for (int pos = 0; pos <= 90; pos += 10) { // Increase in steps of 5°
    for (int i = 0; i < 6; i++) {
      if (servosToRun[i] != -1) {
        board1.setPWM(servosToRun[i], 0, angleToPulse(pos));
      } }
    delay(DELAYTIME);
  }
  delay(1000); // Stay at 90° for 1 sec
  // Move servos from 90° → 0°
  for (int pos = 90; pos >= 0; pos -= 10) { // Decrease in steps of 5°
    for (int i = 0; i < 6; i++) {
      if (servosToRun[i] != -1) {
        board1.setPWM(servosToRun[i], 0, angleToPulse(pos));
      } }
    delay(DELAYTIME);
  } }
void ForNumbers() {
  int servosToRun[6] = { 1, 2, 3, 4, -1, -1 }; // Servos for number prefix (dots 3, 4, 5, 6)
  // Move servos from 0° to 90°
  for (int pos = 0; pos <= 90; pos += 10) {
    for (int i = 0; i < 6; i++) {
      if (servosToRun[i] != -1) {
        board1.setPWM(servosToRun[i], 0, angleToPulse(pos));
      } }
```

```
    delay(DELAYTIME); }
  delay(1000); // Hold at 90° for 1 second
  // Move servos from 90° to 0°
  for (int pos = 90; pos >= 0; pos -= 10) {
    for (int i = 0; i < 6; i++) {
      if (servosToRun[i] != -1) {
        board1.setPWM(servosToRun[i], 0, angleToPulse(pos));
      } }
    delay(DELAYTIME);
  } }
void ForAlphbets() {
  int servosToRun[6] = { 5, -1, -1, -1, -1, -1 }; // Servo for letter prefix (dot 6
  // Move servo from 0° to 90°
  for (int pos = 0; pos <= 90; pos += 10) {
    for (int i = 0; i < 6; i++) {
      if (servosToRun[i] != -1) {
        board1.setPWM(servosToRun[i], 0, angleToPulse(pos));
      }}
    delay(DELAYTIME);  }
  delay(1000); // Hold at 90° for 1 second
  // Move servo from 90° to 0°
  for (int pos = 90; pos >= 0; pos -= 10) {
    for (int i = 0; i < 6; i++) {
      if (servosToRun[i] != -1) {
        board1.setPWM(servosToRun[i], 0, angleToPulse(pos));
```

```
    }
   }
   delay(DELAYTIME);
  }
}
#endif
```

GLOBAL:

```
#ifndef GLOBAL_H
#define GLOBAL_H
#include <LiquidCrystal.h>
#include "SoftwareSerial.h"
#include "DFRobotDFPlayerMini.h"
#include <Adafruit_PWMServoDriver.h>
Adafruit_PWMServoDriver board1 = Adafruit_PWMServoDriver(0x40);
#define SERVOMIN  125  // Pulse width for 0°
#define SERVOMAX  625  // Pulse width for 180°
#define DELAYTIME 10   // Delay for smooth movement
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
const int numMembers = 2;
char names[numMembers][20] = {"NEHA", "KAVYA"};
char accountNumbers[numMembers][13] = {"123456789012",
"234567890123"};
char* AtmPin[numMembers] = {"1212", "2323"};
int accountBalances[numMembers] = {10000, 12000};
```

```
int eepromAddress = 0;
enum Account
{
  MAIN_MENU,
  FACE_CHECK,
  ATM_PIN,
  ENTER_ATM_PIN,
  FORGOT_PIN,
  AMOUNT,
  BALANCE_ENQUIRY,
  WITH_DRAWAL
};
Account currentState = MAIN_MENU;
int newBalance = 0;
char f_data, pre_data = 'X';
#define EEPROM_BALANCE_START_ADDR 0
static const uint8_t PIN_MP3_TX = 50; // Connects to module's RX
static const uint8_t PIN_MP3_RX = 51; // Connects to module's TX
SoftwareSerial softwareSerial(PIN_MP3_RX, PIN_MP3_TX);
const int button1Pin = 48;
const int button2Pin = 47;
const int button3Pin = 46;
const int button4Pin = 45;
const int button5Pin = 44;
const int button6Pin = 43;
```

```
const int button7Pin = 42;
const int button8Pin = 41;
const int button9Pin = 40;
const int button10Pin = 39;
const int confirmButtonPin = 49;
int pos;
char Send = 'x';
int btn1State;
int btn2State;
int btn3State;
int btn4State;
int btn5State;
int btn6State;
int btn7State;
int btn8State;
int btn9State;
int btn10State;
bool flag, amtflag = false;
int confirmState;
char brailleInput[4];
char lastbraile[3];
int index = 0;
int brailleIndex = 0;
char Option = 'x';
int currentCard = -1;
```

```
int originalBalance;
String ReceivedString = "x";
char currentChar = 'x';
char PresentChar = 'x';
int count = 0;
char Atm_Pin = 'x';
int IntDigit;
char digitChar;
int digit;
String NumberString;
bool atmpin = false;
void buttonRead();
void brailleclear();
char* currentNumber;
int Numberlength;
int Namelength;
char* currentName;
#endif
```

REFERENCES

[1] Canser Bilir and Adil Dö¸seyen, “Optimization of atm and branch cash operations using an integrated cash requirement forecasting and cash optimization model,” Business & Management Studies: An International Journal, vol. 6, no. 1, pp. 237–255, 2018.

[2] Fazilet Ozer, Ismail Hakki Toroslu, Pinar Karagoz, and Ferhat Yucel, “Dynamic programming solution to atm cash replenishment optimization problem,” in International Conference on Intelligent Computing & Optimization. Springer, 2018, pp. 428–437.

[3] Hossein Abbasimehr, Mostafa Shabani, and Mohsen Yousefi, “An optimized model using lstm network for demand forecasting,” Computers & Industrial Engineering, p. 106435, 2020

[4] Mithat Zeydan and Sümeyra Kayserili, “A rule-based decision support approach for site selection of automated teller machines (atms),” Intelligent Decision Technologies, vol. 13, no. 2, pp. 161–175, 2019.

[5] S Poorzaker Arabani and H Ebrahimpour Komleh, “The optimization of forecasting atms cash demand of iran banking network using lstm deep recursive neural network,” Journal of Operational Research In Its Applications (Applied Mathematics)-Lahijan Azad University, vol. 16, no. 3, pp. 69–88, 2019.

[6] Sarveswararao Vangala and Ravi Vadlamani, “Atm cash demand forecasting in an indian bank with chaos and deep learning,” arXiv preprint arXiv:2008.10365, 2020.

[7] Seyma Batı and Didem Gözüpek, “Joint optimization of cash management and routing for new-generation automated teller machine networks,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017.

[8] Soodabeh Poorzaker Arabani and Hosein Ebrahimpour Komleh, “The improvement of forecasting atms cash demand of iran banking network using convolutional neural network,” Arabian Journal for Science and Engineering, vol. 44, no. 4, pp. 3733–3743, 2019.

[9] Valeria Platonova, Elena Gubar, and Saku Kukkonen, “Heuristic optimization for multi-depot vehicle routing problem in atm network model,” in Advances in Dynamic Games, pp. 201–228. Springer, 2020.

[10] Yaoguo Li, Huiye Sun, Chen Zhang, and Guohui Li, “Sites selection of atms based on particle swarm optimization,” in 2009 International Conference on Information Technology and Computer Science. IEEE, 2009, vol. 2, pp. 526–530.