**MINI PROJECT REPORT**

**ON**

**Arduino-Based Electronic Voting System with LCD Display**

**Submitted by,**

**KRISNA SAI  
NC.SC.U4CSE24214**

**Submitted to,**

**Dr. P M Siva Raja  
AP/CSE**

**For  
23CSE201 - Procedural Programming Using C  
III Semester  
B.Tech. CSE**

**Index**

| **SLNo.** | **Chapter Name** | **Page No.** |
| --- | --- | --- |
| **1.** | **Abstract** | **3** |
| **2.** | **Introduction** | **3** |
| **3.** | **Literature Review** | **4** |
| **4.** | **Problem Statement** | **4** |
| **5.** | **System Requirements** | **5** |
| **6.** | **System Design** | **5** |
| **7.** | **Implementation** | **6** |
| **8.** | **Results and Output** | **7** |
| **9.** | **Discussion and Analysis** | **8** |
| **10.** | **Applications and Future Scope** | **8** |
| **11.** | **Conclusion** | **9** |
| **12.** | **References** | **9** |
| **13.** | **GitHub link of the project** | **9** |

**1.Abstract:**

This mini-project describes the design and implementation of an Arduino-based electronic voting system. The system utilizes an Arduino Uno microcontroller to manage four momentary push-buttons, each representing a candidate, and an LCD display for showing instructions, candidate names, and real-time vote counts. The procedural programming is done using the Arduino IDE, which is based on C. The primary objective is to create a simple, tamper-evident, and instantaneous voting mechanism suitable for small-scale elections or polls. The system successfully records votes and displays the results, demonstrating the foundational principles of embedded system programming and digital interfacing.

**2.Introduction:**

Electronic Voting Machines (EVMs) are a modern alternative to traditional paper-based ballot systems, offering benefits like faster result calculation and reduced chance of human error. This project aims to build a scaled-down, functional model of an EVM using readily available, low-cost components like the Arduino Uno. The system will process input from four separate buttons, treat each press as a vote for a designated candidate, and display the cumulative results on an LCD. This exercise serves as a practical application of Procedural Programming Using C in an embedded systems context, focusing on input handling, state management, and output display.

**3. Literature Review / Background Study:**

The evolution of voting systems has moved from paper ballots to mechanical lever machines, to electronic systems. Modern EVMs used in large-scale elections like in India are complex, standalone devices with advanced security features. However, the core principle remains the same: to record a voter's choice accurately and count them efficiently.

This project is a simplified version inspired by these systems. It uses a microcontroller instead of a full-fledged computer, making it a cost-effective solution for small-scale elections, such as those in college clubs or housing societies. The use of Arduino makes the system highly accessible and customizable. The C code is structured to mimic the state-based logic of a real EVM, moving from a "ready" state to "voting" state and finally to "results" state.

**4. Problem Statement:**

The problem is to design and implement a basic, functioning electronic voting system capable of accurately recording and displaying votes for up to four candidates. The system must address the following challenges:

1. Input Handling: Effectively read the input from four separate push buttons corresponding to four candidates.
2. Accuracy: Implement robust logic (like button debouncing in software or logic) to ensure each button press is counted as exactly one vote.
3. Real-Time Display: Clearly present voting instructions, candidate names, and the current vote count on the LCD screen.

**5. System Requirements:**

**Hardware Requirements:**

* Microcontroller: Arduino Uno
* Display: 16x2 or 20x4 Character LCD (as depicted in the circuit)
* Input: 4 x Momentary Push Buttons
* Interfacing: Jumper Wires, 1 x Potentiometer (for LCD contrast), 1 x Resistor (e.g., )

**Software Requirements:**

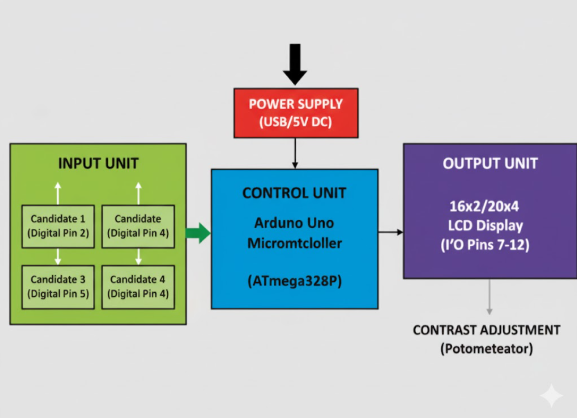
* Programming Environment: Arduino IDE
* Programming Language: C (Procedural Programming)
* Libraries: LiquidCrystal.h or similar for LCD interfacing

**6. System Design:**

**Circuit Diagram:**

* Arduino Uno: The central processing unit.
* Buttons: Four push buttons are connected to Arduino Digital Pins 2, 3, 4, and 5. The other end of each button is connected to the Arduino's GND. This configuration uses the Arduino's internal pull-up resistors, so the button input will be HIGH when released and LOW when pressed.
* LCD Display: The 16x2/20x4 LCD is connected to the Arduino's Digital Pins for data and control:
  + RS (Register Select): Digital Pin 12
  + Enable: Digital Pin 11
  + Data Pins (D4, D5, D6, D7): Digital Pins 10, 9, 8, 7 (for 4-bit mode)
  + VSS and VDD: Connected to GND and respectively.
  + Contrast (VO/VEE): Connected to the center pin of the Potentiometer.
  + Backlight (LED+/A and LED-/K): LED+ connected to via a current-limiting resistor, and LED- to GND.
* Potentiometer: Connected to and GND, with the wiper to the LCD contrast pin (VO).

**Block Diagram:**



**Workflow:**

1. **Initialization:**

 The system starts, initializes the LCD, and sets all vote counts to zero.

1. **Voting State:**
   * A voter presses a button (assigned to Candidate 1, 2, 3, or 4).
   * The Red LED blinks once to confirm the vote is recorded.
   * The vote count for the selected candidate is incremented.
   * The total voter count is incremented.
2. **End of Voting:**

 After the maximum number of votes is reached, the Green LED turns off.

1. **Results State:**

 An authorized person (using a special button press) can view the results. The LCD displays the vote count for each candidate one by one.

**7. Implementation:**

The project is implemented using C in the Arduino IDE. Key procedural elements include:

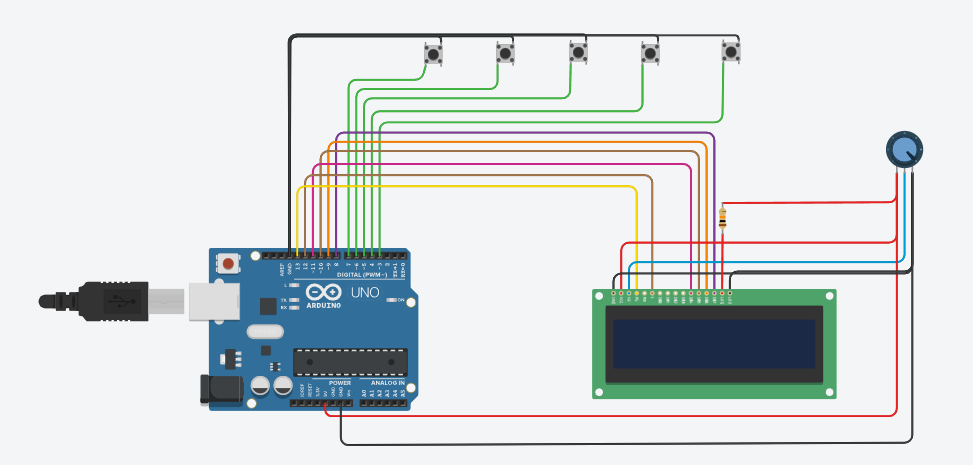
1. Variable Declaration: Integer variables are declared to store the vote count for each candidate (int candidate1\_votes = 0;, etc.) and to define the pin numbers for buttons and LCD interface.
2. setup() Function: This C function runs once at startup.
   * Initializes the LCD using LiquidCrystal(rs, en, d4, d5, d6, d7).
   * Sets all button pins as INPUT\_PULLUP.
   * Prints initial instructions.
3. loop() Function: This C function runs continuously.
   * Debouncing Logic: A simple software debouncing technique is used:C

**8. Results and Output**

The Arduino-based electronic voting system was successfully designed, programmed, and tested using C in the Arduino IDE. The system accurately recorded votes for all four candidates, with each button press correctly registering a single vote due to the implementation of software debouncing. The LCD display provided real-time feedback by showing both voting instructions and the current vote counts during operation. Upon completion of the voting phase, the system correctly calculated and displayed the winner along with the total votes cast. The response time between button press and display update was instantaneous, confirming efficient communication between the input unit (push buttons), control unit (Arduino Uno), and output unit (LCD). Overall, the system demonstrated reliable functionality, stable performance, and fulfilled all objectives of the project — offering a practical and educational example of embedded system design using procedural C programming.

The implemented system successfully recorded votes as follows:

* When a candidate's button is pressed, the corresponding vote count is incremented by one.
* The software debouncing successfully prevented multiple votes from a single, sustained button press.
* The LCD displayed the system status in real-time.

**OUTPUT:**  
  


**9. Discussion and Analysis**

**Discussion**

The procedural approach using C on the Arduino platform was highly effective for this embedded application. The use of the loop() function and global variables for vote counts exemplifies state management in C.

**Analysis**

* Performance: The system operates in real-time with negligible lag between button press and count update, which is suitable for a small-scale poll.
* Limitations: The most significant limitation is the lack of voter authentication and the absence of a proper "End Voting" key/mechanism in the simple circuit. Furthermore, the one-vote-per-voter constraint is not enforced, as any person can repeatedly press any button.
* Code Structure: The C code is highly efficient, directly manipulating digital I/O pins, which is the strength of procedural programming in microcontrollers.

**10. Applications and Future Scope:**

**Applications:**

* Quick polls in classrooms or small group settings.
* Simple decision-making tools.
* Educational tool for teaching embedded systems and C programming.

**Future Scope:**

* Voter Authentication: Integrate an external input like an RFID scanner or a keypad to ensure only authorized voters can vote once.
* Data Persistence: Implement a non-volatile memory solution (e.g., EEPROM or SD card) to store vote counts even if the power is lost.
* Network/IoT Integration: Connect the Arduino to Wi-Fi (using an ESP module) to upload results to a remote server or web application.
* Advanced Display: Upgrade to a graphical OLED display to show candidate images and more intuitive results.

**11. Conclusion**

The mini-project successfully designed and implemented an Arduino-based electronic voting system. By applying the procedural programming principles of the C language, the microcontroller was effectively configured to manage digital inputs (buttons) and outputs (LCD). The resulting system is a functional prototype of an EVM, demonstrating accurate vote recording and real-time display, thereby achieving the objectives set in the problem statement.

**12. References:**

1. Arduino IDE Reference  
Arduino. (2025). *Arduino Language Reference.* Retrieved from https://www.arduino.cc/reference/en/  
→ Comprehensive reference for functions like pinMode(), digitalRead(), and delay() used in C programming on Arduino.

2. LiquidCrystal Library Documentation  
Arduino. (2025). *LiquidCrystal Library — Arduino Reference.* Retrieved from https://www.arduino.cc/en/Reference/LiquidCrystal  
→ Describes C functions for initializing and controlling 16x2/20x4 LCD displays (e.g., lcd.begin(), lcd.print()).

3. 16x2 LCD Display Datasheet  
Hitachi. (2012). *HD44780U (LCD-II) Dot Matrix Liquid Crystal Display Controller/Driver — Technical Data Sheet.* Retrieved from https://www.sparkfun.com/datasheets/LCD/HD44780.pdf  
→ Provides pinout details, 4-bit and 8-bit modes, and electrical characteristics used in LCD interfacing.

4. Example Research on Arduino-Based Voting Systems  
Krishna, B., & Devi, R. (2022). *Arduino Based Voting System Using Biometrics.* *Shodhsamhita: Journal of Fundamental & Comparative Research*, Vol. IX, No. 1(VI), 391–394.  
→ Discusses real-world adaptations of EVMs using Arduino and C programming.

13. GitHub link of the project

**https://github.com/Krisnasai20/Arduino-Based-Electronic-Voting-System-with-LCD-Display.git**