Digital Combination Lock

An Embedded System Project Presented to

Engr. Niño Val Tyrone Young

A faculty of the Computer Studies and Engineering Department

Asian Development foundation College

Tacloban City, Philippines

In Fulfillment of the Requirement for

CPE 421 – Embedded Systems

By:

Cris Darwin Alonzo

Arth Joachim T. Cabarce

Nyla Ericka S. Cadite

Clarence Coral Esquierdo

Yency Shane P. Nguyen

2025

2. Overview / Abstract

This project presents the design and implementation of an embedded digital combination lock system utilizing the Arduino microcontroller. Digital combination locks are essential electronic security devices employed to restrict access to systems until a specific, validated numeric sequence is correctly provided. This system leverages a 4x4 matrix keypad for user input and employs EEPROM (Electrically Erasable Programmable Read-Only Memory) for the persistent, non-volatile storage of the secret access code, enabling robust security and dynamic code management.

The core operational principles involve continuous keypad scanning, the execution of sophisticated embedded security logic for input validation, and precise output control via a two-channel relay module. Real-time user feedback is provided through an integrated LCD screen, a buzzer, and dual LED indicators. Notably, the system incorporates a thematic simulation—emulating a Counter-Strike 2 C4 bomb—to enhance engagement. A successful code entry triggers the green LED and a distinct success tone, signifying system disarming, while code failure or timer expiration activates the red LED and a prolonged alarm sound, displaying the message "!!! EXPLODED !!!". This integrated embedded system serves as both a functional access control prototype and an effective educational model for demonstrating fundamental concepts in digital security, input-output interfacing, and controlled actuation.

Features include:

Password System

- 4-digit password protection
- Default password: 0000 (stored in EEPROM)
- Password saved permanently in EEPROM (remains after power off)
- Hidden input display (shows “****” when typing)
- Option to change password by pressing "A" key
- Password change requires entering old PIN first, then the new PIN

Countdown Timer

- 10-second countdown timer
- The countdown timer is only initiated and starts running after the user enters the first digit (0-9) of the PIN.
- Displays remaining time in seconds on the LCD

- Emits beep sound every 1 second using buzzer
- Countdown automatically resets after explosion or disarms
- Countdown pauses during password change to prevent false explosion

Explosion Simulation

- If no correct password within 10 seconds → explodes automatically
- If wrong password entered → immediate explosion
- Explosion triggers:
    - Red LED ON, Green LED OFF
    - Long buzzer alarm sound
    - LCD shows: "!!! EXPLODED !!!"
    - System auto-resets after explosion

Disarm System

- Entering the correct password disarms the system
    - Green relay turns ON, red relay turns OFF
    - Plays a success tone using the buzzer
    - LCD shows: "DISARMED!"
- System then resets to wait for new activation

EEPROM Integration

- Saves updated password to memory
- Loads last saved password on startup
- Uses EEPROM validation (reverts to default if data is invalid)

Keypad Controls

- | 0–9 | Enter digits for PIN |
- | * | Clear current input |
- | A | Change password mode |
- | B, C, D, # | Reserved (can be customized later) |

LCD Display

- Displays system status messages clearly:
- "SYSTEM ARMED"
- "Enter PIN:"

- "Time Left: Xs"
- "DISARMED!"
- "!!! EXPLODED !!!"
- "PIN UPDATED!"
- "Wrong Old PIN!"
- Automatically clears and refreshes screens for each phase

3. Objectives

The primary objective is to design and implement a digital combination lock system that fulfills the requirements for Embedded Systems.

Specific objectives include:

1. Implement secure access control using a 4-digit password system.
2. Utilize an Arduino microcontroller to process user input and control system outputs.
3. Integrate EEPROM storage for persistent code retention, ensuring the password remains after power-off.
4. Provide real-time user feedback through an LCD screen, LEDs, and a buzzer.
5. Incorporate a countdown timer and an explosion simulation to enhance the security model and C4 aesthetic. The countdown timer will be activated by the first digit input by the user.
6. Allow for dynamic password updates via the keypad.

4. System Components

4.1. Hardware

The hardware components are the physical elements responsible for input, processing, and output.

Components	Functions
Arduino UNO	The central Microcontroller Unit (MCU); processes input, executes logic, and controls all outputs.
4x4 Matrix Keypad	Primary user input device for entering numeric codes (PINs).
Two-Channel Relay Module	Simulates the locking/unlocking mechanism (access granted/denied).

LED (Red & Green)	Provides immediate visual feedback (Red for denied/explosion, Green for success/disarmed).
LCD Screen (I2C)	Displays user prompts, status messages ("Enter PIN:", "DISARMED!" "!!! EXPLODED !!!"), and the countdown time.
Buzzer	Emits auditory signals for user feedback (beeps for timer/alerts for confirmation/error).
Batteries	Supplies power to the system, enabling portability.
Stripboard	Platform for mounting and connecting components for semi-permanent circuit construction.
Cardboard Box	Houses the assembly, serving as protection and simulating the miniature C4 bomb exterior.
Solid Wire/Jumper Wires	Facilitates electrical connections between modules and the Arduino.

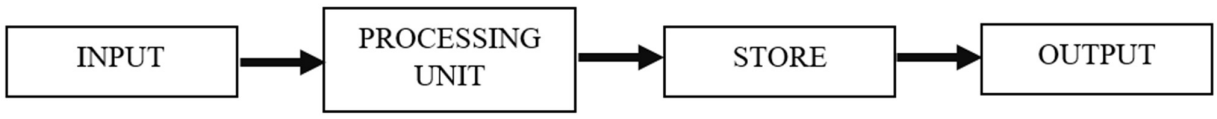
4.2. Software

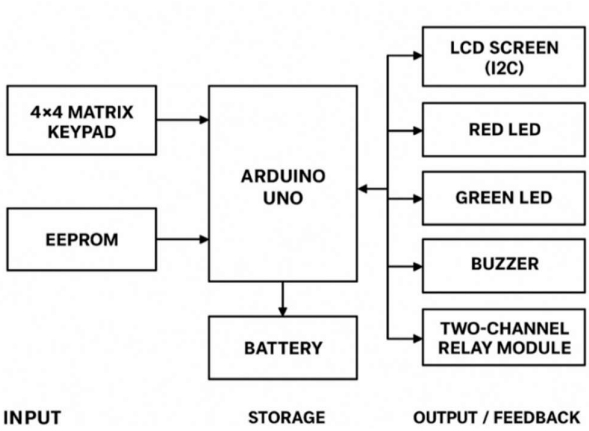
The software primarily consists of the **Arduino Code** (written in C/C++ based on the snippets) and the necessary **libraries**:

- **Arduino Code (Sketch):** Contains the core logic for keypad scanning, password validation, EEPROM read/write, countdown management, and controlling the relays, LEDs, and buzzer.
- **EEPROM.h Library:** Used for storing the password persistently so it remains saved even after power-off.
- **Wire.h and LiquidCrystal_I2C.h Libraries:** Used to communicate with and control the LCD screen via the I2C interface.

5. System Diagram

5.1 Block Diagram

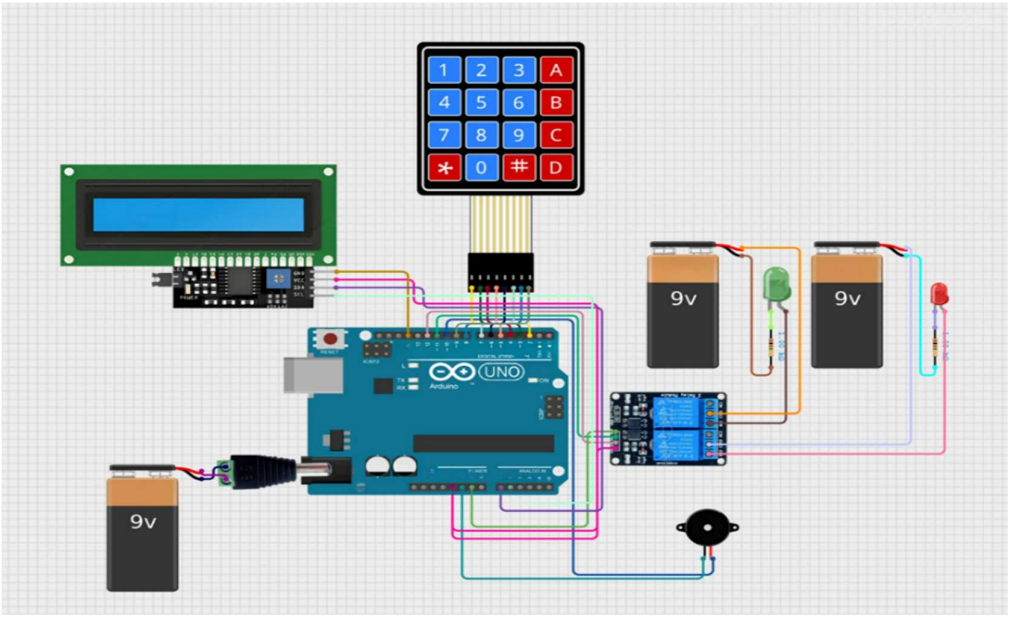




The **Arduino UNO** is positioned at the center of the block diagram, serving as the main processing unit. Directional arrows indicate the flow of data, signals, and power between the Arduino and external components.

- **Input:** 4x4 Matrix Keypad (Input to Arduino)
- **Store:** EEPROM (Internal/External storage for password)
- **Output/Feedback:** LCD Screen I2C, Red LED, Green LED, Buzzer, Two-Channel Relay Module (Output from Arduino)
- **Power:** Battery (Input to system)

5.2 Schematic Diagram



6. Methodology

a. System Initialization/Activation:

The process begins immediately upon power-up, focusing on establishing the security state and loading the credentials.

- The system loads the last saved password from the EEPROM upon startup (or reverts to the default "0000" if invalid data is found).
- The LCD displays "SYSTEM ARMED" and "Time Left: 10s".
- The system initializes in a **'REST' state**; the countdown timer is **NOT** running. The LCD displays the prompt to "Enter PIN:".
- The countdown timer will only begin simultaneously when the first digit of the PIN is entered by the user.

b. Input Reading (Keypad Scanning):

The system dedicates resources to detecting user input via the physical interface.

- The Arduino constantly monitors the 4x4 matrix keypad using row-column logic to detect key presses.
- As the user types the PIN (0-9), the input is displayed as hidden input ("****") on the LCD. The '*' key clears the current input.

c. Arduino Processing (Security Logic):

The core logic runs two simultaneous processes: handling the keypad input and managing the timer.

- **Keypad Handler (handleKeypadInput()):** This function is the primary driver for input-based state changes.
    - If the system is in the initial 'REST' state, the first press of a digit ('0'-'9') will activate the 10-second countdown, starting the timer from that moment.
    - When a key is detected, the entered digit is appended to the temporary input string.
    - If the user presses the '#' key (usually designated as the "Enter" or "Submit" key, or implicitly after 4 digits), the system triggers the comparison.
    - The full entered sequence is compared, digit by digit, to the password retrieved from the EEPROM.
- **Countdown Handler (handleCountdown()):** This function is time-critical and runs independently of the keypad waiting.
    - It only executes if the countdown has been activated by a user input.

- It decrements the time displayed on the LCD every second.
- A synchronous *beep sound* is emitted from the buzzer with each decrement, creating a sense of urgency.
- The countdown can only be stopped or paused by a successful action (like disarming or entering the change password mode)

d. Output Display & Actuation (Disarm or Explosion):

The system's final action is determined by the outcome of the security logic comparison or the timer's expiration. This involves physical output actuation.

Condition	System State	LCD Message	LED/Relay Response	Buzzer Response
Correct PIN Entered	Disarmed (Access Granted)	"DISARMED"	Green LED On, (Safe) Red LED Off	Success Tone (Short, distinct melody)
Wrong PIN Entered	Exploded (Access Denied)	"!!!EXPLODED!!!"	Red LED On, Green LED OFF	Long Alarm Sound
Timer Expires (10s)	Exploded (Time Out)	"!!!EXPLODED!!!"	Red LED On, Green LED OFF	Long Alarm Sound

e. Password Change Function:

This is a privileged operation that requires pre-authentication and temporarily halts the security clock.

- Pressing the **'A' key** enters the password change mode.
- The user must first enter the **old PIN**. If correct, they are prompted for the new 4-digit PIN.
- The new password is then saved to the **EEPROM**. The **countdown timer is paused** during this operation

f. System Reset:

The system ensures that after any terminal event (Disarm or Explosion), it is immediately ready for a new cycle.

- **Auto-Reset:** After the consequence of the event is displayed and enacted (e.g., the buzzer sounds and the LEDs are set), the system automatically resets its internal state.
- **Return to Armed:** It returns to the **"**SYSTEM ARMED**"** state, clears the keypad input buffer, and restarts the 10-second countdown, awaiting a new user activation.

7. Code

The code, written for the Arduino platform, includes definitions for pin numbers, configurations (Relay, Keypad, Password), and the main logic functions:

- **Configuration (setup()):** Initializes pins as outputs (Relays, Buzzer, LED) or inputs, initializes the LCD, and calls loadPassword() to retrieve the stored PIN.
- **Main Loop (loop()):** Continuously calls handleKeypadInput() to check for user keys and conditionally calls handleCountdown() to manage the timer and trigger the explosion if time runs out.
- **Password Management:**
  - savePassword(): Writes the new password to the EEPROM.
  - loadPassword(): Reads the PIN from EEPROM and validates it, reverting to "1234" (or "0000" as per features) if invalid.
  - changePassword(): Manages the sequence of entering the old PIN and saving the new one.
- **System Response:**
  - disarmSystem(): Triggered by a correct PIN; turns the Green relay ON, displays "DISARMED!," plays a success tone, and resets the system.
  - explodeNow(): Triggered by a wrong PIN or timeout; turns the Red relay ON, displays "!!! EXPLODED !!!," plays a long alarm tone, and resets the system.
  - resetSystem(): Clears status flags and resets the countdown timer variables.

8. Testing & Results

The system's functionality was rigorously validated through a series of iterative function checks and comprehensive end-to-end scenario simulations. Testing began with Basic PIN Entry to ensure the disarmSystem() function reliably activated upon entering the default password ("0000") or a correctly updated password. The critical Failure Modes were also tested, confirming that the explodeNow() function was correctly triggered both when an incorrect password was entered and when the 10-second countdown timer expired.

Verification and Observed Behavior

A key focus was Persistent Storage, where testing verified that the updated password remained saved in the EEPROM after the system was powered off and restarted, thereby validating data integrity. The complex Password Change Flow was meticulously tested, requiring the user to press 'A', enter the old PIN, and then the new PIN, with correct "PIN UPDATED!" and error "Wrong Old PIN!" messages displayed for confirmation. Throughout these tests, Feedback and Timing were monitored to ensure the accuracy of the 1-second interval beeps from the buzzer, the correct operation of the LEDs, and the synchronized LCD messages.

The Observed Behavior confirmed the successful Final Output of the LCD, which displayed the armed status and the current countdown (e.g., "Time Left: 4s / Enter PIN:"). The Keypad Functionality was verified, allowing reliable user input. Both primary scenarios were confirmed: Disarm Simulation correctly resulted in the Green LED ON and the LCD displaying "DISARMED!", while the Explosion Simulation upon failure (wrong code or timeout) correctly showed the Red LED ON and the LCD message "!!! EXPLODED !!!".

Troubleshooting

- **Keypad Reading:** Ensuring the keypad library correctly read all 16 keys without debouncing issues.
- **EEPROM Integrity:** Verifying that data read from the EEPROM was not corrupt and that the validation logic correctly reverted to the default PIN if necessary.
- **Relay Actuation:** Ensuring the Arduino pin voltage was sufficient to reliably trigger the relay module switches for the ON/OFF states.
- **I2C LCD:** Resolving potential issues with the I2C address or library compatibility for displaying the status messages.

9. Limitations

The system, as a functional prototype, has several limitations:

- **Input Security:** The current 4-digit PIN is relatively weak; it could be easily brute-forced.
- **Security Logic:** The system offers only one try; if the password is wrong, it immediately "explodes." In a real-world scenario, multiple tries with a lockout period would be more practical.
- **Hardware and Casing:** The use of a cardboard box makes the system vulnerable to external damage and environmental factors (humidity, physical impact).
- **Power Requirements:** The battery limits the operational time and may need frequent replacement or recharging.
- **Aesthetic:** The exposed wiring and components on the prototype board, visible during assembly, do not represent a polished, industrial-grade lock.

10. Recommendations

Based on the current prototype, the following improvements are recommended:

- **Enhanced Security:** Increase the password length (e.g., 6-8 digits) and implement a lockout feature (e.g., waiting 60 seconds after 3 failed attempts) to prevent rapid brute-forcing.
- **User Interface:** Replace the simple LEDs with an RGB LED to provide a wider range of visual status cues (e.g., yellow for "lockout") and use a more durable enclosure (e.g., 3D-printed or metal) for a professional look.
- **Remote/Wireless Access:** Integrate a Bluetooth (HC-05) or Wi-Fi (ESP8266) module to allow the user to arm, disarm, or change the password remotely via a mobile app.

- **Audio/Feedback:** Use a more advanced audio module to play more complex tones or even voice prompts, instead of just the simple buzzer tones.

11. Conclusion

This project successfully designed and implemented a digital combination lock embedded system using an Arduino UNO microcontroller. The system effectively integrates a 4x4 matrix keypad for input, EEPROM for persistent password storage, and an LCD/LED/buzzer triad for immediate feedback. It functions as a secure access control prototype while providing an engaging, game-themed simulation of a C4 bomb with a countdown timer and explosion logic. The core operational concepts—keypad scanning, output control via relays, and embedded security—were successfully demonstrated, fulfilling the requirements for an embedded systems project.