KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
COE 381 – MICROPROCESSORS

GROUP 9 PRESENTS
EMERGENCY ALERT SYSTEM FOR PATIENTS
BY

NYARKO PRINCE EDWIN (1825622)
OPOKU DUODU MICHAEL (1826022)
OWUSU PRINCE (1827122)
ABAZAAMI MALEX AYITINYA (1812122)
QUAYE MARK TETTEH (1827622)
MENSAH EDWARD SACKEY (1824422)
SOBBIN FLAVIO NANA BADU (1828322)
ALPHATSON COBBINA SIAW (1828222)
AWUSAH EMMANUEL ELIKPLIM (1819122)
AGYAPONG ALBERT YEBOAH (1814422)

TABLE OF CONTENTS

INTRODUCTION

1.1 Project Overview

The Emergency Alert System for Patients is a dedicated communication solution designed to rapidly signal emergencies across multiple hospital wards. It provides a reliable way to alert staff and emergency responders, ensuring timely action during critical situations.

1.1.1 Objectives

- Rapid Notification: Enable instant communication of emergency events.
- Ease of Use: Ensure a simple interface that requires minimal training.
- Reliability: Use hardware design practices to minimize false triggers.
- Scalability: Provide a foundation for future integration with network systems and centralized monitoring.

1.2 Problem Statement

Hospitals require a dependable method to instantly signal emergencies across various wards to ensure a swift and coordinated response. Existing emergency systems can be overly complex, difficult to operate, or slow in transmitting alerts, which may delay critical interventions. Moreover, some systems may generate false alarms due to sensitivity issues or poor design, leading to alarm fatigue among staff.

This project addresses these challenges by developing a simple, user-friendly, and robust Hospital Emergency Alert System. The system is designed to:

- **Minimize Simultaneous Triggers:** Implementing hardware and software debouncing techniques reduces the likelihood of unintended alerts.
- **Ensure Rapid Response:** By offering immediate visual and audible notifications, the system enables staff to respond quickly to emergencies.
- **Enhance Usability:** The straightforward design requires minimal training, ensuring that even non-technical staff can operate it effectively.
- **Promote System Reliability:** Automatic reset features ensure the system is always ready for subsequent alerts, reducing downtime and maintenance.
- **Facilitate Future Scalability:** The design allows for future integration with advanced monitoring systems and network-based alert mechanisms, providing a pathway for continuous improvement.

By tackling these issues, the Hospital Emergency Alert System aims to improve patient safety and operational efficiency within hospital environments.

METHODOLOGY

The development of the Hospital Emergency Alert System follows a structured process to ensure a robust and reliable design:

2.1 Requirement Analysis

- **Stakeholder Input:** Identify key requirements from hospital staff regarding alert response times, user interface simplicity, and reliability.
- **System Specifications:** Define technical requirements such as response speed, power consumption, and environmental conditions.

2.1.1 Hardware Design & Component Selection

- **Component Evaluation:** Choose suitable components (Arduino UNO, 16x2 LCD, push buttons, and active buzzer) based on performance, cost, and availability.
- **Circuit Design:** Develop a schematic diagram that shows interconnections between components, ensuring proper wiring, voltage levels, and pin assignments.

2.1.2 Software Development

- Firmware Implementation: Write and test code for input detection, debounce logic, and alert triggering. Emphasis is placed on creating clear visual and audible notifications.
- Integration Testing: Validate the functionality of the software by simulating button presses and ensuring that the LCD and buzzer respond correctly.

2.1.3 Prototyping and Assembly

- **Breadboarding:** Assemble the initial prototype on a breadboard to test the system's functionality in a controlled environment
- **Component Testing:** Evaluate each hardware component (e.g., button responsiveness, LCD display clarity, buzzer volume) before final assembly.

2.1.4 System Testing and Validation

- **Functional Testing:** Conduct tests to ensure that the system responds correctly to emergency alerts across all wards.
- **Debounce and Reset Verification:** Validate that the debounce logic minimizes false triggers and that the system resets properly after each alert.
- **User Testing:** Engage potential users (hospital staff) in testing the system for usability and clarity of the alerts

2.1.5 Iteration and Improvement

- **Feedback Collection:** Gather feedback from testing phases to identify areas for improvement.
- **Refinement:** Update the design and software based on testing results and user feedback to optimize performance and reliability.
- **Documentation:** Update all documentation and schematics to reflect the final design and any modifications made during testing.

2.1.6 Future Enhancements

- **Scalability Plans:** Outline steps for integrating additional features such as wireless communication, remote monitoring, and event logging.
- **Continuous Improvement:** Establish a process for periodic review and updates to ensure the system remains effective and incorporates emerging technologies.

2.2 Hardware Components

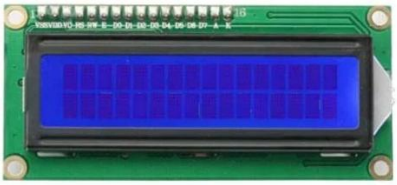
2.2.1 Use of Components

1. Arduino UNO Microcontroller

Role: Acts as the central processing unit (CPU) of the system.
Usage:

- Reads inputs from the push buttons.
- Processes the logic for triggering alerts.
- Controls the LCD display and buzzer based on user input.
- Implements debounce logic to avoid false triggers.

2. Adafruit 16x2 LCD Display



Role: Provides visual feedback.
Usage:

- Displays the system status, such as "Emergency Alert System Ready" on startup.
- Shows emergency messages like "EMERGENCY ALERT!" along with the specific ward number when an alert is triggered.
- Resets to the "System Ready" state after an alert sequence.

3. Push Button

Role: Serve as the user interface for triggering emergency alerts.
Usage:

- Each button represents a different hospital ward.
- When pressed, the corresponding button sends a LOW signal to the Arduino, triggering an emergency alert.
- Configured with internal pull-up resistors to ensure stable and reliable input readings.

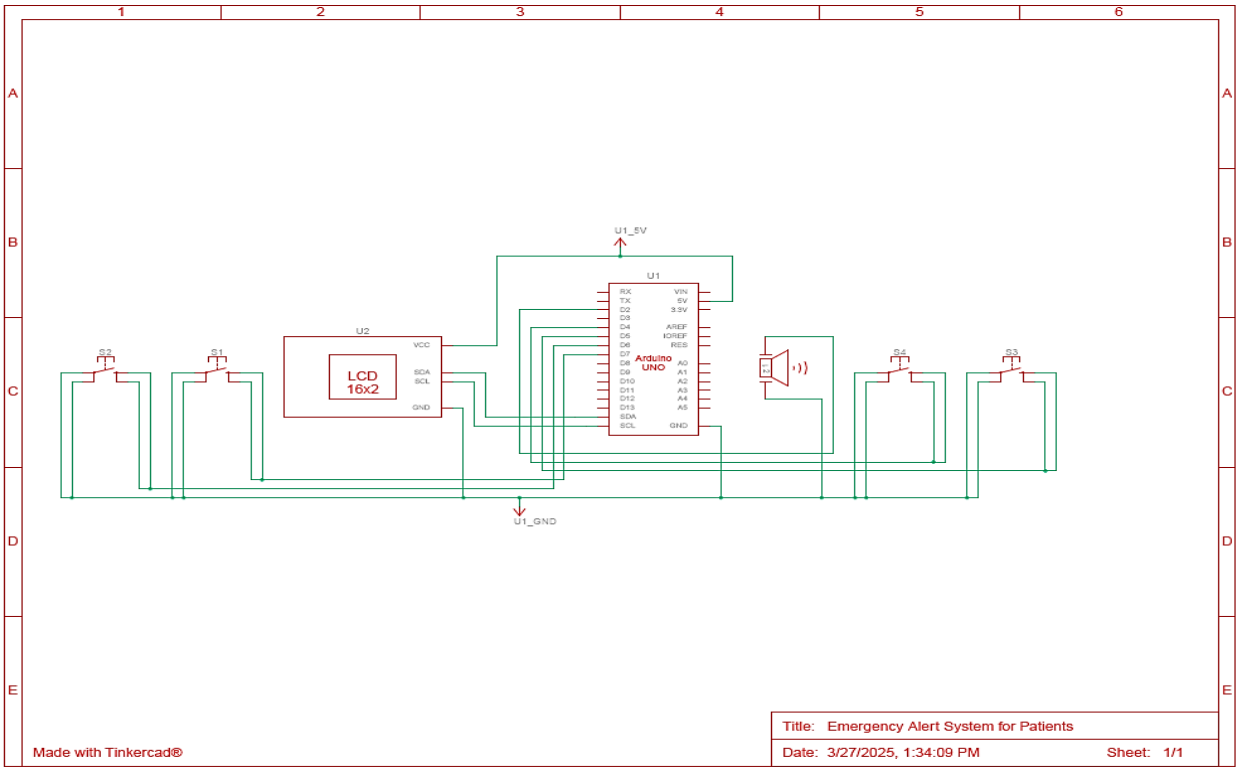
4. Buzzer

- **Role:** Provides audible alerts
- **Usage:**
  - Emits an alternating tone pattern (high tone followed by low tone) to signal an emergency.
  - The buzzer is activated when any ward button is pressed, repeating the tone sequence several times to ensure the alert is noticed.
  - Controlled by the Arduino to synchronize with the visual alerts on the LCD.

2.2.2 Pin Configuration

Component	Arduino Pin	Notes
Ward 1 Button	Digital Pin 7	Input with internal pull-up resistor
Ward 2 Button	Digital Pin 6	Input with internal pull-up resistor
Ward 3 Button	Digital Pin 5	Input with internal pull-up resistor
Ward 4 Button	Digital Pin 4	Input with internal pull-up resistor
Buzzer	Digital Pin 2	Configured as output
LCD Display	I2C Interface	Uses Address 0x20

2.3 Circuit Diagram



2.4 System Functionality

2.4.1 Initialization

Input Setup

- **Emergency Button Configuration:**
  Each of the four emergency buttons is connected to a specific digital pin on the Arduino. During initialization, these pins are set up as inputs
- **Internal Pull-Up Resistors:**
  The buttons are configured with internal pull-up resistors, which ensure that each button reads a HIGH signal when unpressed. When a button is pressed, it pulls the pin voltage LOW. This configuration:
  - Reduces the likelihood of erratic behavior due to electrical noise.
  - Provides a clear and stable signal for detecting an actual button press.
- **Debounce Considerations:**
  Although the hardware configuration with pull-up resistors enhances signal stability, the code also implements a debounce delay in the main loop. This delay further prevents false triggering by filtering out rapid fluctuations that may occur when the button contacts physically bounce.

Output Setup

- **Buzzer Configuration:**
  The buzzer, used for audible alerts, is connected to a designated digital output pin. During initialization:

- The pin is set as an output to enable the Arduino to control the buzzer.
- The system ensures that the buzzer remains off by default, avoiding any unintended sounds during startup.

- **LCD Display Initialization via I2C:**
  The 16x2 LCD display is connected using the I2C interface, which minimizes wiring complexity and allows for efficient communication.
  - **Communication Setup:**
    The I2C protocol is initialized, establishing a communication link between the Arduino and the LCD.
  - **Display Configuration:**
    The display is set to the correct dimensions (16 columns by 2 rows) and is cleared of any previous data.

  - **Startup Message:**

  Immediately after initialization, the LCD is programmed to show the message **"Emergency Alert System Ready"**. This message serves several purposes:
    - **Operational Confirmation:** Indicates to users that the system has successfully booted and is functioning correctly.
    - **User Guidance:** Provides clear feedback that the system is in standby mode, ready to detect any emergency input.
    - **System Check:** Acts as a visual test to ensure that the LCD and its communication via I2C are properly working.

  - **Timing and Delays**
    - Startup Delay:

    After displaying the initial message, a short delay (e.g., 2 seconds) is introduced. This delay allows users to read the startup confirmation and ensures that all hardware components have stabilized.
    - Transition to Normal Operation:

    Once the delay expires, the LCD is cleared to prepare it for real-time updates during emergency alerts. This clearing action transitions the system from the initialization phase to normal operational mode, where it begins actively monitoring for button presses.

2.4.2 Key Design Considerations

- **Debounce Delay:** A short delay is implemented to mitigate false triggers due to mechanical bounce from the buttons.
- **Self-Resetting Mechanism:** The system automatically reverts to the standby state after each alert.
- **User-Friendly Interface:** The clear LCD display and simple button layout facilitate ease of use for hospital staff.
- **Low Power Consumption:** The design is optimized for energy efficiency to ensure continuous operation.
- **Scalability:** The system architecture supports future enhancements such as wireless communication, event logging, and centralized monitoring.

2.5 Code Explanation and Implementation

This section outlines the logic behind the code that powers the system:

2.5.1 Initialization Phase

- Library and LCD Setup:

The system uses an LCD library to initialize a 16x2 display. A startup message is shown to confirm that the system is active.

- Pin Configuration:

Emergency buttons are set up as inputs with internal pull-up resistors, while the buzzer is set up as an output for sound generation.

2.5.2 Main Loop

- Monitoring Inputs:

The code continuously checks the state of the four buttons. A button press (LOW signal) triggers an emergency alert.

- Debounce Handling:

A brief delay in the main loop prevents multiple triggers from a single press due to contact bounce.

2.5.3 Alert Handling

- User Feedback:

When an alert is triggered, the LCD updates to display an emergency message along with the ward identifier.

- Audible Notification:

The buzzer produces a repeating sequence of high and low tones to ensure the alert is noticed.

- System Reset:

After the alert sequence, the system pauses briefly before resetting the display to the default "System Ready" state.

CONCLUSION

The Hospital Emergency Alert System offers a streamlined, reliable solution for emergency communication in hospital settings. Its combination of visual and audible alerts, user-friendly design, and robust hardware practices ensure that critical messages are delivered swiftly and effectively. Future enhancements such as remote monitoring and data logging can further extend its capabilities, making it a scalable solution for modern healthcare environments.

REFERENCES

[1] Arduino Documentation: Arduino Reference

[2] I2C LCD Display Guide: Adafruit 16x2 LCD with I2C Backpack

[3] Debounce Techniques: All About Circuits on Button Debouncing

[4] IEEE Standards: IEEE Standards