



Smart Emergency Room Detector using Priority Encoder (Arduino-based)



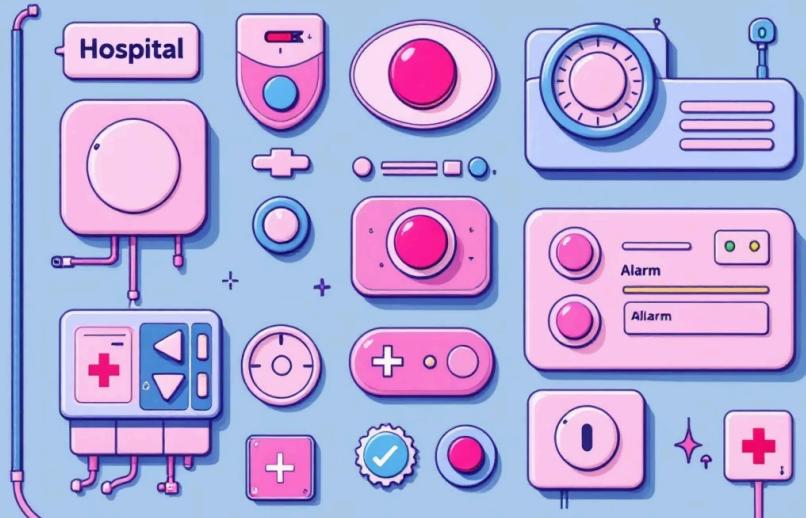
Introduction: Prioritizing Urgent Medical Attention

In critical hospital environments, numerous emergencies can arise simultaneously, demanding immediate and precise attention. This project introduces an Arduino-based [Smart Emergency Room Detector](#) designed to identify and prioritize these urgent signals, ensuring medical staff are alerted to the most critical situations first.

Project Objectives: Clarity in Crisis

Detect Multiple Emergencies

To design a system capable of receiving various emergency inputs from different sources within a hospital setting.



Prioritize Critical Alerts

To ensure that when multiple emergencies occur simultaneously, the system automatically identifies and highlights the most critical one.



Working Theory: The Priority Encoder Logic



Input Simulation

Each push button represents a distinct emergency level (High, Medium, Low).



Output Indication

LEDs (Red, Yellow, Green) indicate the status of each emergency level.



Arduino as Encoder

The Arduino continuously scans inputs and acts as the priority encoder.



Priority Logic

When multiple buttons are pressed, the system prioritizes (e.g., Red > Yellow > Green), activating only the highest priority LED.

This intelligent logic ensures only the most critical emergency is visually highlighted at any given time.

Essential Components for the Build

Arduino UNO	1	Main controller for logic processing
Breadboard	1	Prototyping and circuit connections
LEDs (Red, Yellow, Green)	3	Visual output indicators for emergency levels
Push Buttons	3	Simulate emergency inputs (High, Medium, Low)
Resistors (220Ω, 10kΩ)	6	Current limiting for LEDs and pull-down for buttons
Jumper Wires	As required	Interconnecting all components

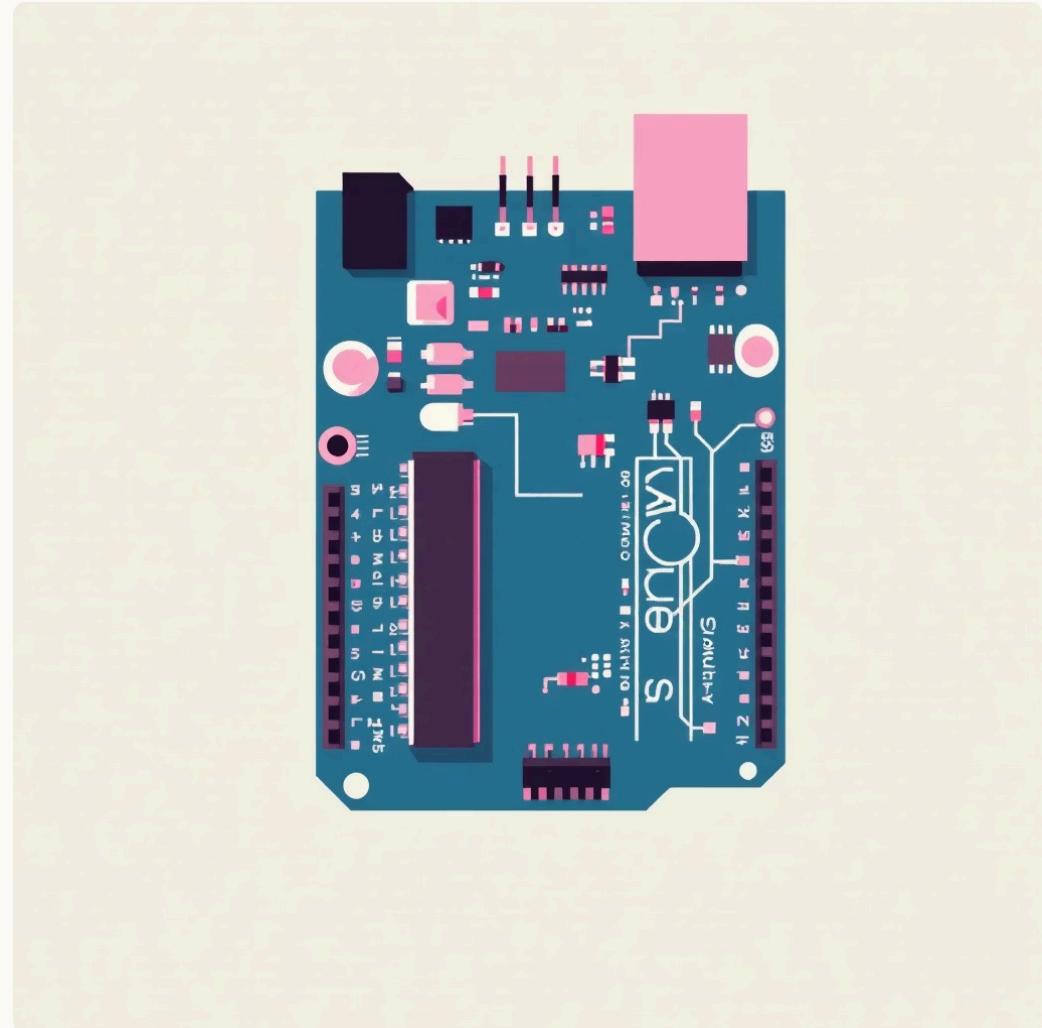
Circuit Explanation: Wiring the Priority System

The circuit is designed for clear input detection and priority-based output:

- Each push button is connected to a dedicated digital input pin (2, 3, 4) on the Arduino, using 10k Ω pull-down resistors to ensure stable readings.
- LEDs are connected to output pins (8, 9, 10), each with a 220 Ω resistor to protect them from excessive current.

The Arduino's programming logic will then handle the priority:

- If button 1 (High priority) is pressed, the **Red LED** illuminates.
- Else, if button 2 (Medium priority) is pressed, the **Yellow LED** illuminates.
- Else, if button 3 (Low priority) is pressed, the **Green LED** illuminates.



Arduino Code: The Logic Behind Priority

The Arduino sketch is concise yet powerful, continuously monitoring inputs and updating outputs based on predefined priority rules.

```
// Define input pins for buttons
const int highPriorityButton = 2;
const int mediumPriorityButton = 3;
const int lowPriorityButton = 4;

// Define output pins for LEDs
const int redLED = 8; // High priority
const int yellowLED = 9; // Medium priority
const int greenLED = 10; // Low priority

void setup() {
    pinMode(highPriorityButton, INPUT);
    pinMode(mediumPriorityButton, INPUT);
    pinMode(lowPriorityButton, INPUT);

    pinMode(redLED, OUTPUT);
    pinMode(yellowLED, OUTPUT);
    pinMode(greenLED, OUTPUT);
}

void loop() {
    // Turn off all LEDs initially
    digitalWrite(redLED, LOW);
    digitalWrite(yellowLED, LOW);
    digitalWrite(greenLED, LOW);

    // Check for high priority first
    if (digitalRead(highPriorityButton) == HIGH) {
        digitalWrite(redLED, HIGH);
    }
    // If high priority is not active, check for medium priority
    else if (digitalRead(mediumPriorityButton) == HIGH) {
        digitalWrite(yellowLED, HIGH);
    }
    // If neither high nor medium is active, check for low priority
    else if (digitalRead(lowPriorityButton) == HIGH) {
        digitalWrite(greenLED, HIGH);
    }
}
```



Results and Output: Instant Prioritized Alerts



Single Emergency Activation

Pressing any single emergency button causes its corresponding LED to glow.



Highest Priority Prevails

If multiple buttons are pressed, only the LED corresponding to the highest priority will illuminate.



Effective Simulation

The system effectively simulates a priority-based emergency response, proving its design objectives.

Diverse Applications of Priority Encoders

Hospital Emergency Alerts

Crucial for rapidly identifying and escalating critical patient needs.

Fire & Disaster Systems

Prioritizing evacuation routes and critical resource allocation in emergencies.

Industrial Process Monitoring

Alerting operators to the most critical system faults in manufacturing or control processes.

Smart Automation

Managing tasks and device responses based on user-defined importance.

Conclusion: Efficient Priority Management

This project successfully demonstrates the practical implementation of a priority encoder using Arduino, offering an **efficient and clear method** for handling multiple simultaneous signals. By ensuring that the most critical condition is always brought to immediate attention, this system can significantly improve response times and effectiveness in high-stakes environments.

