Imitation of Automated Lift Control System Using STM32 MCU

Objective:
Design and develop an embedded system that simulates an automated lift (elevator) control system using an STM32 microcontroller. The system should allow users to request floors via push-buttons, display the current floor position on a 7-segment display, and provide real-time status updates on an LCD screen. A relay will simulate lift movement, and a buzzer will generate alerts for door open/close, floor arrival, and emergency conditions.

Description:
The system will function as an imitation of a multi-floor lift control panel, allowing users to request floor movement via push-buttons. The 7-segment display will show the current floor, while the LCD screen will provide detailed status updates such as "Moving Up," "Moving Down," "Door Open," or "Overload". A relay will simulate the lift motor, and a buzzer will provide alerts when the lift arrives at a floor or when an emergency stop is activated.

Expected Outcome:
A fully functional imitation of an automated lift control system using an STM32 MCU, where users can request floor movement, receive real-time feedback on a 7-segment display and LCD, and experience simulated lift operations with relay and buzzer alerts.

Step 1 : 7-segment display

Code :

```
#include "main.h"

void display_digit(uint8_t digit);

const uint8_t seg_digit_map[4] = {
  0b1000000, // 0
  0b1111001, // 1
  0b0100100, // 2
  0b0110000  // 3
};

int main(void)
{
  HAL_Init();
  __HAL_RCC_GPIOC_CLK_ENABLE();

  GPIO_InitTypeDef GPIO_InitStruct = {0};
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|
                        GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  while (1)
  {
    for(uint8_t i = 0; i < 4; i++) {
      display_digit(i);
```

```
      HAL_Delay(1000);
    }
  }
}

void display_digit(uint8_t digit)
{
  uint8_t seg = seg_digit_map[digit];
  for(int i = 0; i < 7; i++) {
    HAL_GPIO_WritePin(GPIOC, (1 << i), (seg >> i) & 1);
  }
}
```

output : https://drive.google.com/file/d/114yNM7GLGRonQ-nZaDYE-wPpRGtaTJIZ/view?usp=sharing

Step 2 : LCD and Switch

code :

```
#include "main.h"
#include <string.h>

// LCD Pins (PC7 to PC12)
#define LCD_RS_Pin GPIO_PIN_7
#define LCD_EN_Pin GPIO_PIN_8
#define LCD_D4_Pin GPIO_PIN_9
#define LCD_D5_Pin GPIO_PIN_10
#define LCD_D6_Pin GPIO_PIN_11
#define LCD_D7_Pin GPIO_PIN_12
#define LCD_PORT GPIOC

// Switches (PA0, PA1, PA4, PA6)
#define SW_G_Pin GPIO_PIN_0
#define SW_1_Pin GPIO_PIN_1
#define SW_2_Pin GPIO_PIN_4
#define SW_EM_Pin GPIO_PIN_6
#define SW_PORT GPIOA

char floor_msg[16];
int emergency_mode = 0;

void lcd_pulse_enable(void) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_EN_Pin, GPIO_PIN_SET);
  HAL_Delay(1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_EN_Pin, GPIO_PIN_RESET);
  HAL_Delay(1);
}

void lcd_send_nibble(uint8_t nibble) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_D4_Pin, (nibble >> 0) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D5_Pin, (nibble >> 1) & 1);
```

```
  HAL_GPIO_WritePin(LCD_PORT, LCD_D6_Pin, (nibble >> 2) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D7_Pin, (nibble >> 3) & 1);
  lcd_pulse_enable();
}

void lcd_send_cmd(uint8_t cmd) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_RS_Pin, GPIO_PIN_RESET);
  lcd_send_nibble(cmd >> 4);
  lcd_send_nibble(cmd & 0x0F);
  HAL_Delay(2);
}

void lcd_send_data(uint8_t data) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_RS_Pin, GPIO_PIN_SET);
  lcd_send_nibble(data >> 4);
  lcd_send_nibble(data & 0x0F);
  HAL_Delay(2);
}

void lcd_clear(void) {
  lcd_send_cmd(0x01);
  HAL_Delay(2);
}

void lcd_init(void) {
  HAL_Delay(40);
  lcd_send_nibble(0x03); HAL_Delay(5);
  lcd_send_nibble(0x03); HAL_Delay(5);
  lcd_send_nibble(0x03); HAL_Delay(1);
  lcd_send_nibble(0x02); HAL_Delay(1);

  lcd_send_cmd(0x28); // 4-bit, 2-line
  lcd_send_cmd(0x0C); // Display ON
  lcd_send_cmd(0x06); // Entry mode
  lcd_clear();
}

void lcd_print(char* str) {
  while (*str) lcd_send_data(*str++);
}

void update_floor_display(const char* floor_label) {
  lcd_clear();
  sprintf(floor_msg, "Current Floor: %s", floor_label);
  lcd_print(floor_msg);
}

void show_emergency_message(void) {
  lcd_clear();
  lcd_print("Emergency Stop");
}
```

```
int main(void)
{
  HAL_Init();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  GPIO_InitTypeDef GPIO_InitStruct = {0};

  // LCD Pins
  GPIO_InitStruct.Pin = LCD_RS_Pin | LCD_EN_Pin |
                        LCD_D4_Pin | LCD_D5_Pin |
                        LCD_D6_Pin | LCD_D7_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(LCD_PORT, &GPIO_InitStruct);

  // Switch Pins
  GPIO_InitStruct.Pin = SW_G_Pin | SW_1_Pin | SW_2_Pin | SW_EM_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(SW_PORT, &GPIO_InitStruct);

  lcd_init();
  update_floor_display("G");

  while (1)
  {
    if (emergency_mode) {
      continue; // Do nothing permanently
    }

    if (HAL_GPIO_ReadPin(SW_PORT, SW_EM_Pin) == GPIO_PIN_RESET) {
      emergency_mode = 1;
      show_emergency_message();
      continue;
    }

    if (HAL_GPIO_ReadPin(SW_PORT, SW_G_Pin) == GPIO_PIN_RESET) {
      update_floor_display("G");
      HAL_Delay(300);
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_1_Pin) == GPIO_PIN_RESET) {
      update_floor_display("1");
      HAL_Delay(300);
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_2_Pin) == GPIO_PIN_RESET) {
      update_floor_display("2");
      HAL_Delay(300);
    }
  }
}
```

output : https://drive.google.com/file/d/1jd0CqeIhQpkdw-ktVgCA4aRTpTW5sR-x/view?usp=sharing

Step 3 : Adding buzzer to the entire circuit

code :

```
#include "main.h"
#include <string.h>

// LCD Pins
#define LCD_RS_Pin GPIO_PIN_7
#define LCD_EN_Pin GPIO_PIN_8
#define LCD_D4_Pin GPIO_PIN_9
#define LCD_D5_Pin GPIO_PIN_10
#define LCD_D6_Pin GPIO_PIN_11
#define LCD_D7_Pin GPIO_PIN_12
#define LCD_PORT GPIOC

// Switches
#define SW_G_Pin GPIO_PIN_0
#define SW_1_Pin GPIO_PIN_1
#define SW_2_Pin GPIO_PIN_4
#define SW_EM_Pin GPIO_PIN_6
#define SW_PORT GPIOA

// Buzzer
#define BUZZER_Pin GPIO_PIN_1
#define BUZZER_PORT GPIOB

const uint8_t seg_digit_map[3] = {
  0b1000000, // G
  0b1111001, // 1
  0b0100100  // 2
};

int emergency_mode = 0;
char floor_msg[20];

void lcd_pulse_enable(void) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_EN_Pin, GPIO_PIN_SET);
  HAL_Delay(1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_EN_Pin, GPIO_PIN_RESET);
  HAL_Delay(1);
}

void lcd_send_nibble(uint8_t nibble) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_D4_Pin, (nibble >> 0) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D5_Pin, (nibble >> 1) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D6_Pin, (nibble >> 2) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D7_Pin, (nibble >> 3) & 1);
```

```
  lcd_pulse_enable();
}

void lcd_send_cmd(uint8_t cmd) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_RS_Pin, GPIO_PIN_RESET);
  lcd_send_nibble(cmd >> 4);
  lcd_send_nibble(cmd & 0x0F);
  HAL_Delay(2);
}

void lcd_send_data(uint8_t data) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_RS_Pin, GPIO_PIN_SET);
  lcd_send_nibble(data >> 4);
  lcd_send_nibble(data & 0x0F);
  HAL_Delay(2);
}

void lcd_clear(void) {
  lcd_send_cmd(0x01);
  HAL_Delay(2);
}

void lcd_init(void) {
  HAL_Delay(40);
  lcd_send_nibble(0x03); HAL_Delay(5);
  lcd_send_nibble(0x03); HAL_Delay(5);
  lcd_send_nibble(0x03); HAL_Delay(1);
  lcd_send_nibble(0x02); HAL_Delay(1);

  lcd_send_cmd(0x28); // 4-bit, 2-line
  lcd_send_cmd(0x0C); // Display ON
  lcd_send_cmd(0x06); // Entry mode
  lcd_clear();
}

void lcd_print(char* str) {
  while (*str) lcd_send_data(*str++);
}

void update_floor_display(const char* floor_label) {
  lcd_clear();
  sprintf(floor_msg, "Current Floor: %s", floor_label);
  lcd_print(floor_msg);
}

void display_digit(uint8_t digit_index) {
  uint8_t seg = seg_digit_map[digit_index];
  for (int i = 0; i < 7; i++) {
    HAL_GPIO_WritePin(GPIOC, (1 << i), (seg >> i) & 1);
  }
}
```

```
void show_emergency_message(void) {
  lcd_clear();
  lcd_print("Emergency Stop");
}

int main(void)
{
  HAL_Init();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  GPIO_InitTypeDef GPIO_InitStruct = {0};

  // LCD
  GPIO_InitStruct.Pin = LCD_RS_Pin | LCD_EN_Pin |
                        LCD_D4_Pin | LCD_D5_Pin |
                        LCD_D6_Pin | LCD_D7_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LCD_PORT, &GPIO_InitStruct);

  // 7-Segment
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|
                        GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  // Switches
  GPIO_InitStruct.Pin = SW_G_Pin | SW_1_Pin | SW_2_Pin | SW_EM_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(SW_PORT, &GPIO_InitStruct);

  // Buzzer
  GPIO_InitStruct.Pin = BUZZER_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  HAL_GPIO_Init(BUZZER_PORT, &GPIO_InitStruct);
  HAL_GPIO_WritePin(BUZZER_PORT, BUZZER_Pin, GPIO_PIN_RESET);

  lcd_init();
  update_floor_display("G");
  display_digit(0);

  while (1)
  {
    if (emergency_mode) {
      HAL_GPIO_WritePin(BUZZER_PORT, BUZZER_Pin, GPIO_PIN_SET);
      continue;
    }

    if (HAL_GPIO_ReadPin(SW_PORT, SW_EM_Pin) == GPIO_PIN_RESET) {
      emergency_mode = 1;
```

```
      show_emergency_message();
      display_digit(0); // freeze at current (optional)
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_G_Pin) == GPIO_PIN_RESET) {
      update_floor_display("G");
      display_digit(0);
      HAL_Delay(300);
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_1_Pin) == GPIO_PIN_RESET) {
      update_floor_display("1");
      display_digit(1);
      HAL_Delay(300);
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_2_Pin) == GPIO_PIN_RESET) {
      update_floor_display("2");
      display_digit(2);
      HAL_Delay(300);
    }
  }
}
```

output : https://drive.google.com/file/d/1FBcQSkWMQcyoyhXzHSitKPk-4EKET5U1/view?usp=sharing

Step 4 : Moving instructions

code :

```
#include "main.h"
#include <string.h>

// LCD Pins
#define LCD_RS_Pin GPIO_PIN_7
#define LCD_EN_Pin GPIO_PIN_8
#define LCD_D4_Pin GPIO_PIN_9
#define LCD_D5_Pin GPIO_PIN_10
#define LCD_D6_Pin GPIO_PIN_11
#define LCD_D7_Pin GPIO_PIN_12
#define LCD_PORT GPIOC

// Switches
#define SW_G_Pin GPIO_PIN_0
#define SW_1_Pin GPIO_PIN_1
#define SW_2_Pin GPIO_PIN_4
#define SW_EM_Pin GPIO_PIN_6
#define SW_PORT GPIOA

// Buzzer
#define BUZZER_Pin GPIO_PIN_1
#define BUZZER_PORT GPIOB

const uint8_t seg_digit_map[3] = {
  0b1000000, // G
  0b1111001, // 1
  0b0100100  // 2
```

```
};

int current_floor = 0; // 0 = G, 1 = 1st, 2 = 2nd
int emergency_mode = 0;
char lcd_line[20];

void lcd_pulse_enable(void) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_EN_Pin, GPIO_PIN_SET);
  HAL_Delay(1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_EN_Pin, GPIO_PIN_RESET);
  HAL_Delay(1);
}

void lcd_send_nibble(uint8_t nibble) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_D4_Pin, (nibble >> 0) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D5_Pin, (nibble >> 1) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D6_Pin, (nibble >> 2) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D7_Pin, (nibble >> 3) & 1);
  lcd_pulse_enable();
}

void lcd_send_cmd(uint8_t cmd) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_RS_Pin, GPIO_PIN_RESET);
  lcd_send_nibble(cmd >> 4);
  lcd_send_nibble(cmd & 0x0F);
  HAL_Delay(2);
}

void lcd_send_data(uint8_t data) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_RS_Pin, GPIO_PIN_SET);
  lcd_send_nibble(data >> 4);
  lcd_send_nibble(data & 0x0F);
  HAL_Delay(2);
}

void lcd_clear(void) {
  lcd_send_cmd(0x01);
  HAL_Delay(2);
}

void lcd_set_cursor(uint8_t row, uint8_t col) {
  lcd_send_cmd(0x80 | (row == 1 ? 0x40 : 0x00) | col);
}

void lcd_print(char* str) {
  while (*str) lcd_send_data(*str++);
}

void lcd_init(void) {
  HAL_Delay(40);
  lcd_send_nibble(0x03); HAL_Delay(5);
  lcd_send_nibble(0x03); HAL_Delay(5);
```

```
  lcd_send_nibble(0x03); HAL_Delay(1);
  lcd_send_nibble(0x02); HAL_Delay(1);

  lcd_send_cmd(0x28); // 4-bit, 2-line
  lcd_send_cmd(0x0C); // Display ON
  lcd_send_cmd(0x06); // Entry mode
  lcd_clear();
}

void display_digit(uint8_t digit_index) {
  uint8_t seg = seg_digit_map[digit_index];
  for (int i = 0; i < 7; i++) {
    HAL_GPIO_WritePin(GPIOC, (1 << i), (seg >> i) & 1);
  }
}

void show_emergency_message(void) {
  lcd_clear();
  lcd_set_cursor(0, 0);
  lcd_print("Emergency Stop");
  lcd_set_cursor(1, 0);
  lcd_print("System Locked!");
}

void update_lift_status(int target_floor) {
  if (target_floor == current_floor) return;

  // Door Closing
  lcd_set_cursor(1, 0);
  lcd_print("Door Closing    ");
  HAL_Delay(1000);

  // Moving Up / Down
  lcd_set_cursor(1, 0);
  lcd_print((target_floor > current_floor) ? "Moving Up      " : "Moving Down    ");
  HAL_Delay(3000);  // Simulated lift delay

  // Door Opening
  lcd_set_cursor(1, 0);
  lcd_print("Door Opening    ");
  HAL_Delay(1000);

  // Floor Display Update
  lcd_clear();
  lcd_set_cursor(0, 0);
  sprintf(lcd_line, "Current Floor: %s", (target_floor == 0 ? "G" : (target_floor == 1 ? "1" : "2")));
  lcd_print(lcd_line);
  lcd_set_cursor(1, 0);
  lcd_print("Door Open       ");

  display_digit(target_floor);
  current_floor = target_floor;
```

```
}

int main(void)
{
  HAL_Init();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  GPIO_InitTypeDef GPIO_InitStruct = {0};

  // LCD
  GPIO_InitStruct.Pin = LCD_RS_Pin | LCD_EN_Pin |
                        LCD_D4_Pin | LCD_D5_Pin |
                        LCD_D6_Pin | LCD_D7_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LCD_PORT, &GPIO_InitStruct);

  // 7-Segment
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|
                        GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  // Switches
  GPIO_InitStruct.Pin = SW_G_Pin | SW_1_Pin | SW_2_Pin | SW_EM_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(SW_PORT, &GPIO_InitStruct);

  // Buzzer
  GPIO_InitStruct.Pin = BUZZER_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  HAL_GPIO_Init(BUZZER_PORT, &GPIO_InitStruct);
  HAL_GPIO_WritePin(BUZZER_PORT, BUZZER_Pin, GPIO_PIN_RESET);

  lcd_init();
  lcd_set_cursor(0, 0);
  lcd_print("Current Floor: G");
  lcd_set_cursor(1, 0);
  lcd_print("Door Open");
  display_digit(0);

  while (1)
  {
    if (emergency_mode) {
      HAL_GPIO_WritePin(BUZZER_PORT, BUZZER_Pin, GPIO_PIN_SET);
      continue;
    }

    if (HAL_GPIO_ReadPin(SW_PORT, SW_EM_Pin) == GPIO_PIN_RESET) {
      emergency_mode = 1;
```

```
      show_emergency_message();
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_G_Pin) == GPIO_PIN_RESET) {
      update_lift_status(0);
      HAL_Delay(300);
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_1_Pin) == GPIO_PIN_RESET) {
      update_lift_status(1);
      HAL_Delay(300);
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_2_Pin) == GPIO_PIN_RESET) {
      update_lift_status(2);
      HAL_Delay(300);
    }
  }
}
```

output : https://drive.google.com/file/d/1NR43rd-NjRfES9EZWU94YV2Nznm8QdU-/view?usp=sharing

Step 5 : Relay to start the system

Final code :

```
#include "main.h"
#include <string.h>

// LCD Pins
#define LCD_RS_Pin GPIO_PIN_7
#define LCD_EN_Pin GPIO_PIN_8
#define LCD_D4_Pin GPIO_PIN_9
#define LCD_D5_Pin GPIO_PIN_10
#define LCD_D6_Pin GPIO_PIN_11
#define LCD_D7_Pin GPIO_PIN_12
#define LCD_PORT GPIOC

// Switches
#define SW_G_Pin GPIO_PIN_0
#define SW_1_Pin GPIO_PIN_1
#define SW_2_Pin GPIO_PIN_4
#define SW_EM_Pin GPIO_PIN_6
#define SW_PORT GPIOA

// Buzzer
#define BUZZER_Pin GPIO_PIN_1
#define BUZZER_PORT GPIOB

// Relay and Master Switch
#define RELAY_Pin GPIO_PIN_0
#define RELAY_PORT GPIOB

#define MASTER_SWITCH_Pin GPIO_PIN_13
#define MASTER_SWITCH_PORT GPIOC

const uint8_t seg_digit_map[3] = {
```

```
  0b1000000, // G
  0b1111001, // 1
  0b0100100  // 2
};

int current_floor = 0; // 0 = G, 1 = 1st, 2 = 2nd
int emergency_mode = 0;
int system_active = 0;
char lcd_line[20];

void lcd_pulse_enable(void) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_EN_Pin, GPIO_PIN_SET);
  HAL_Delay(1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_EN_Pin, GPIO_PIN_RESET);
  HAL_Delay(1);
}

void lcd_send_nibble(uint8_t nibble) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_D4_Pin, (nibble >> 0) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D5_Pin, (nibble >> 1) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D6_Pin, (nibble >> 2) & 1);
  HAL_GPIO_WritePin(LCD_PORT, LCD_D7_Pin, (nibble >> 3) & 1);
  lcd_pulse_enable();
}

void lcd_send_cmd(uint8_t cmd) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_RS_Pin, GPIO_PIN_RESET);
  lcd_send_nibble(cmd >> 4);
  lcd_send_nibble(cmd & 0x0F);
  HAL_Delay(2);
}

void lcd_send_data(uint8_t data) {
  HAL_GPIO_WritePin(LCD_PORT, LCD_RS_Pin, GPIO_PIN_SET);
  lcd_send_nibble(data >> 4);
  lcd_send_nibble(data & 0x0F);
  HAL_Delay(2);
}

void lcd_clear(void) {
  lcd_send_cmd(0x01);
  HAL_Delay(2);
}

void lcd_set_cursor(uint8_t row, uint8_t col) {
  lcd_send_cmd(0x80 | (row == 1 ? 0x40 : 0x00) | col);
}

void lcd_print(char* str) {
  while (*str) lcd_send_data(*str++);
}
```

```
void lcd_init(void) {
  HAL_Delay(40);
  lcd_send_nibble(0x03); HAL_Delay(5);
  lcd_send_nibble(0x03); HAL_Delay(5);
  lcd_send_nibble(0x03); HAL_Delay(1);
  lcd_send_nibble(0x02); HAL_Delay(1);

  lcd_send_cmd(0x28);
  lcd_send_cmd(0x0C);
  lcd_send_cmd(0x06);
  lcd_clear();
}

void display_digit(uint8_t digit_index) {
  uint8_t seg = seg_digit_map[digit_index];
  for (int i = 0; i < 7; i++) {
    HAL_GPIO_WritePin(GPIOC, (1 << i), (seg >> i) & 1);
  }
}

void show_emergency_message(void) {
  lcd_clear();
  lcd_set_cursor(0, 0);
  lcd_print("Emergency Stop");
  lcd_set_cursor(1, 0);
  lcd_print("System Locked!");
}

void update_lift_status(int target_floor) {
  if (target_floor == current_floor) return;

  lcd_set_cursor(1, 0);
  lcd_print("Door Closing    ");
  HAL_Delay(1000);

  lcd_set_cursor(1, 0);
  lcd_print((target_floor > current_floor) ? "Moving Up      " : "Moving Down    ");
  HAL_Delay(3000);

  lcd_set_cursor(1, 0);
  lcd_print("Door Opening    ");
  HAL_Delay(1000);

  lcd_clear();
  lcd_set_cursor(0, 0);
  sprintf(lcd_line, "Current Floor: %s", (target_floor == 0 ? "G" : (target_floor == 1 ? "1" : "2")));
  lcd_print(lcd_line);
  lcd_set_cursor(1, 0);
  lcd_print("Door Open       ");

  display_digit(target_floor);
  current_floor = target_floor;
```

```
}

int main(void)
{
  HAL_Init();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  GPIO_InitTypeDef GPIO_InitStruct = {0};

  // LCD
  GPIO_InitStruct.Pin = LCD_RS_Pin | LCD_EN_Pin |
                        LCD_D4_Pin | LCD_D5_Pin |
                        LCD_D6_Pin | LCD_D7_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LCD_PORT, &GPIO_InitStruct);

  // 7-Segment
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|
                        GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  // Switches
  GPIO_InitStruct.Pin = SW_G_Pin | SW_1_Pin | SW_2_Pin | SW_EM_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(SW_PORT, &GPIO_InitStruct);

  // Buzzer
  GPIO_InitStruct.Pin = BUZZER_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  HAL_GPIO_Init(BUZZER_PORT, &GPIO_InitStruct);
  HAL_GPIO_WritePin(BUZZER_PORT, BUZZER_Pin, GPIO_PIN_RESET);

  // Relay
  GPIO_InitStruct.Pin = RELAY_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  HAL_GPIO_Init(RELAY_PORT, &GPIO_InitStruct);
  HAL_GPIO_WritePin(RELAY_PORT, RELAY_Pin, GPIO_PIN_RESET);  // Initially OFF

  // Master Switch (Blue button on PC13)
  GPIO_InitStruct.Pin = MASTER_SWITCH_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(MASTER_SWITCH_PORT, &GPIO_InitStruct);

  lcd_init();
  lcd_set_cursor(0, 0);
  lcd_print("Press Blue Btn");
```

```
  int last_button_state = 1;

  while (1)
  {
    // Toggle system ON/OFF using master switch (PC13)
    int button_state = HAL_GPIO_ReadPin(MASTER_SWITCH_PORT, MASTER_SWITCH_Pin);
    if (button_state == GPIO_PIN_RESET && last_button_state == GPIO_PIN_SET) {
      system_active = !system_active;
      HAL_GPIO_WritePin(RELAY_PORT, RELAY_Pin, system_active ? GPIO_PIN_SET :
GPIO_PIN_RESET);

      lcd_clear();
      lcd_set_cursor(0, 0);
      lcd_print(system_active ? "System ON" : "System OFF");

      if (!system_active) {
        HAL_GPIO_WritePin(BUZZER_PORT, BUZZER_Pin, GPIO_PIN_RESET);
        while (HAL_GPIO_ReadPin(MASTER_SWITCH_PORT, MASTER_SWITCH_Pin) ==
GPIO_PIN_RESET);
      }

      HAL_Delay(300);
    }
    last_button_state = button_state;

    if (!system_active) continue;

    if (emergency_mode) {
      HAL_GPIO_WritePin(BUZZER_PORT, BUZZER_Pin, GPIO_PIN_SET);
      continue;
    }

    if (HAL_GPIO_ReadPin(SW_PORT, SW_EM_Pin) == GPIO_PIN_RESET) {
      emergency_mode = 1;
      show_emergency_message();
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_G_Pin) == GPIO_PIN_RESET) {
      update_lift_status(0);
      HAL_Delay(300);
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_1_Pin) == GPIO_PIN_RESET) {
      update_lift_status(1);
      HAL_Delay(300);
    } else if (HAL_GPIO_ReadPin(SW_PORT, SW_2_Pin) == GPIO_PIN_RESET) {
      update_lift_status(2);
      HAL_Delay(300);
    }
  }
}
```

final output :https://drive.google.com/file/d/138LR3jnb0zYNw4vuibuKs0rQ_HPMuTe9/view?usp=sharing