A Project Report on

ADVANCED AND SAFE ELEVATOR USING PROTEUS AND KEIL

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Rajiv Gandhi University Of Knowledge Technologies - Nuzvid

Submitted by

TEJO PRIYA BODDU
(N200429)

SRIJA PUDI
(N200574)

Under the Esteemed Guidance of

Mr. M. UMA MAHESWAR RAO, M. Tech.,

APSSDC

Andhra Pradesh State Skill Development Corporation

(Department of Skill Development and Training, Govt of Andhra Pradesh)

SUMMER INTERNSHIP

12-MAY-2025 to 12-JULY-2025

APSSDC

Andhra Pradesh State Skill Development Corporation

(Department of Skill Development and Training, Govt of Andhra Pradesh)

SUMMER INTERNSHIP

CERTIFICATE OF COMPLETION

This is to certify that the work entitled, "**ADVANCED AND SAFE ELEVATOR USING PROTEUS AND KEIL"** is the bonafide work of **B. TEJO PRIYA (N200429)** and **SRIJA PUDI (N200574)** carried out under my guidance and supervision for the **EMBEDDED SYSTEMS** domain in the department of Electronics and Communication Engineering under **APSSDC**. This work is done during the **SUMMER INTERNSHIP** May 2025 – July 2025, under our guidance.

Mr. M.Uma Maheswar Rao

Internal Project Guide,

Department of ECE,

APSSDC

APSSDC

Andhra Pradesh State Skill Development Corporation

(Department of Skill Development and Training, Govt of Andhra Pradesh)

SUMMER INTERNSHIP

CERTIFICATE OF EXAMINATION

This is to certify that the work entitled "**ADVANCED AND SAFE ELEVATOR USING PROTEUS AND KEIL"** is the bonafide work of **B. TEJO PRIYA (N200429)** , **SRIJA PUDI (N200574)** and here by accord our approval of it as a study carried out and presented in a manner required for its acceptance during the Summer Internship, for which it has been submitted. This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusion drawn, as recorded in this thesis. It only signifies the acceptance of this thesis for the purpose for which it has been submitted.

---------- —

Mr. M.Uma Maheswar Rao

Internal Project Guide,

Department of ECE,

APSSDC

APSSDC

Andhra Pradesh State Skill Development Corporation

(Department of Skill Development and Training, Govt of Andhra Pradesh)

SUMMER INTERNSHIP

DECLARATION

B.TEJO PRIYA (N200429) and **SRIJA PUDI (N200574)** hereby declare that the project report entitled "**ADVANCED AND SAFE ELEVATOR USING PROTEUS AND KEIL"** done by us under the guidance of **Mr. M.Uma Maheswar Rao** , Assistant Professor is submitted for the fulfillment of the Summer Internship in Embedded System domain of Bachelor of Technology in Electronics and Communication Engineering during the academic session May 2025 – July 2025, APSSDC.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites are mentioned in the references. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Date: 28-06-2025
Place: Mylavaram

B.TEJO PRIYA (N200429)
P.SRIJA (N200574)

ACKNOWLEDGEMENT

We would like to express our profound gratitude and deep regards to our guide **Mr. M.Uma Maheswar Rao** for his exemplary guidance, monitoring and constant encouragement to us throughout the Summer Internship. We shall always cherish the time spent with him during the course of this work due to the invaluable knowledge gained in the field of reliability engineering. We are extremely grateful for the confidence bestowed in us and entrusting our project entitled "**ADVANCED AND SAFE ELEVATOR USING PROTEUS AND KEIL" .**

We express gratitude to **Mr. M.Uma Maheswar Rao (M.TECH)** and other faculty members for being a source of inspiration and constant encouragement which helped us in completing the project successfully.

Finally, yet importantly, we would like to express our heartfelt thanks to our beloved God and parents for their blessings, our friends for their help and wishes for the successful completion of this project.

TABLE OF CONTENTS

ABSTRACT

This project presents the design and simulation of a Smart and Safe Elevator System using the 8051 microcontroller, implemented through Keil and Proteus. The system enables floor selection using push-button switches and displays the current floor number via a virtual terminal interface. To enhance safety, a fire detection mechanism is simulated using a logic state input. Upon fire alert, the system prioritizes emergency protocols, halting normal elevator operations and activating warnings. The design emphasizes user safety, efficient floor handling, and real-time response to hazardous conditions. This project serves as a practical example of embedded systems application in automation and safety-critical environments, demonstrating the integration of control logic and virtual interfacing in microcontroller-based projects.

CHAPTER 1 INTRODUCTION

1.1 Introduction to Embedded Systems

An **Embedded System** is a specialized computer system designed to perform a dedicated function or a set of specific tasks within a larger mechanical or electrical system. Unlike general-purpose computers, embedded systems are optimized for specific operations and are often designed for real-time computing with constraints on power, performance, and size.

1.1.1 History and Evolution

The concept of embedded systems dates back to the 1960s. The first known embedded system was the Apollo Guidance Computer developed by MIT for NASA's Apollo missions. Over the years, embedded systems have evolved from bulky, standalone controllers to highly compact and powerful modules found in everything from household appliances to aerospace systems. With the advancement of microprocessors and microcontrollers, embedded systems have become more efficient, affordable, and widespread.

1.1.2 Types of Embedded Systems

Embedded systems can be classified based on performance and functional requirements:

- **Real-Time Embedded Systems**: Designed to provide outputs within a defined time frame (e.g., medical devices, automotive ABS).
- **Standalone Embedded Systems**: Operate independently without requiring a host system (e.g., washing machines, calculators).
- **Networked Embedded Systems**: Communicate with other systems via networks like LAN, WAN, or the Internet (e.g., smart meters, IoT devices).
- **Mobile Embedded Systems**: Portable and compact systems embedded in mobile devices (e.g., smartphones, wearable devices).

Classification of Embedded systems based on hardware:

- **Small-scale systems**: Use 8-bit or 16-bit microcontrollers.
- **Medium-scale systems**: Use 16-bit or 32-bit processors and perform more complex operations.
- **Sophisticated systems**: Use powerful processors, FPGAs, or SoCs and may include operating systems like Linux or RTOS.

1.1.3 Applications and Importance

Embedded systems are integral to modern life. They are used in:

- ➢ **Consumer Electronics**: Smart TVs, washing machines, microwaves
- ➢ **Automobiles**: Engine control units, airbag systems, infotainment
- ➢ **Industrial Automation**: PLCs, robotic arms, CNC machines
- ➢ **Medical Devices**: Pacemakers, monitoring equipment
- ➢ **Telecommunications**: Routers, base stations
- ➢ **Aerospace and Defense**: Navigation, control systems, drones

1.1.4 Why Embedded Systems?

The increasing demand for automation, miniaturization, and real-time response makes embedded systems crucial in various industries. Their ability to provide dedicated control with high reliability and efficiency makes them the backbone of modern smart devices.

1.1.5 Advantages

- ❖ High reliability and performance
- ❖ Low power consumption
- ❖ Compact size and low cost
- ❖ Real-time task execution
- ❖ Customizable for specific applications
- ❖ Long lifecycle with minimal maintenance

1.1.6 Tools Used for Development

Developing embedded systems requires both hardware and software tools:

- ➢ **Microcontrollers**: 8051, AVR, PIC, ARM Cortex, STM32, ESP32
- ➢ **Programming Languages**: C, C++, Embedded C, Assembly
- ➢ **IDEs and Compilers**: Keil µVision, MPLAB X, Atmel Studio, Arduino IDE, STM32CubeIDE
- ➢ **Simulation and Debugging Tools**: Proteus, Multisim, QEMU, JTAG/SWD debuggers, Logic analyzers
- ➢ **Real-Time Operating Systems (RTOS)**: FreeRTOS, VxWorks, RTEMS
- ➢ **Communication Interfaces**: UART, SPI, I2C, CAN, USB

1.2 Objectives and Goals of the Project

The primary objective of this project is to design and simulate a **Smart and Safe Elevator System** using the **8051 microcontroller**, incorporating both control logic and safety mechanisms. The project aims to demonstrate the real-world application of embedded systems in automation and public safety.

1.2.1 Main Objectives:

1. To develop a functional elevator simulation that can handle floor navigation using push-button inputs.
2. To display the current floor and system status through a virtual terminal interface.
3. To implement a fire alert mechanism using logic input that triggers emergency response protocols.
4. To utilize Keil for embedded C programming and Proteus for circuit simulation and testing.
5. To ensure safe and efficient vertical movement control based on real-time logic.

1.2.2 Project Goals:

- ➢ Demonstrate real-time system behavior using embedded logic.
- ➢ Provide a prototype model that mimics practical elevator functionalities.
- ➢ Integrate safety features such as emergency stops during fire alerts.
- ➢ Improve understanding of microcontroller interfacing, input handling, and system feedback.
- ➢ Build a foundation for more complex automation systems using 8051 and similar microcontrollers

1.3 Literature Review

The elevator system, though mechanically designed decades ago, has significantly evolved with the integration of microcontroller-based automation and safety features. This literature review explores existing developments in elevator systems, embedded automation, and fire alert integration in embedded platforms.

Numerous research works and projects have emphasized the use of **microcontrollers like the 8051, PIC, and AVR** in designing automated elevator systems. These works show that microcontrollers are ideal for managing elevator logic, button interfacing, and output control due to their cost-effectiveness, flexibility, and ease of programming. For instance, studies have

demonstrated the use of **8051 microcontrollers** in basic two-floor and multi-floor elevator models using push-button switches, relays, and 7-segment displays or LCDs.

The incorporation of safety mechanisms such as **fire detection and emergency shutdowns** has also been a subject of various implementations. In advanced systems, **MQ2 gas sensors** and **temperature sensors** are commonly used to detect smoke or fire presence. However, in simulation-based environments like Proteus, many academic projects have adopted **logic-level inputs** to emulate these hazardous conditions without needing physical sensors. This approach helps in validating system responses during emergencies in a virtual and cost-effective manner.

Simulation tools like **Proteus Design Suite** have become standard in prototyping and validating embedded projects. Coupled with **Keil µVision IDE**, these tools allow students and developers to simulate both code and hardware design, greatly reducing the time and complexity of embedded system development.

Additionally, embedded projects involving elevators often focus on **real-time performance**, **user safety**, **minimal power consumption**, and **system reliability**. Literature also reflects an increasing trend of integrating **communication protocols** (UART, I2C) and **RTOS-based control** in more complex elevator systems, though such approaches are generally reserved for industry-grade applications.

In conclusion, the existing literature provides a strong foundation for implementing a smart and safe elevator system using the 8051 microcontroller. This project builds upon those concepts and contributes by simulating a safe elevator model with integrated fire alert logic, demonstrating a practical and educational approach to embedded system design and safety automation.

1.4 Limitations

While the Smart and Safe Elevator System using 8051 effectively demonstrates core elevator functions and safety features in a simulated environment, it does have certain limitations:

1. **Simulation-Based Only**: The project is limited to software simulation using Proteus and does not include real hardware implementation or physical elevator mechanics.
2. **Fire Alert via Logic State**: Instead of using a real gas or flame sensor, the fire alert feature is simulated using a logic input, which may not fully represent real-world sensor behavior and accuracy.
3. **Limited Number of Floors**: The current implementation is restricted to a small number of floors due to input/output pin limitations of the 8051 microcontroller.

CHAPTER 2 METHODOLOGY

2.1 Methodology

This section explains the step-by-step methodology followed in the design, implementation, and simulation of the Smart and Safe Elevator System using the 8051 microcontroller. The system is developed using embedded programming and simulated using virtual tools to validate its functionality.

2.1.1 Embedded System Architecture

The architecture of this embedded system is structured around the **8051 microcontroller**, which serves as the brain of the elevator control unit. The key components and their roles are:

- ➢ **Input Unit**: Push-button switches represent floor selection (e.g., Ground, First, Second floors). An additional logic state input simulates a fire alert condition.
- ➢ **Processing Unit**: The 8051 microcontroller processes inputs using programmed logic in Embedded C and controls the elevator's behavior accordingly.
- ➢ **Output Unit**: The virtual terminal in Proteus acts as the display system showing the current floor or emergency status. Additional outputs like LEDs or buzzers can be used to represent alerts.
- ➢ **Control Logic**: Decision-making algorithms determine the elevator's movement, halt during fire, and priority handling based on inputs.

2.1.2 Tools Used

The following tools were used for development and simulation:

- ❖ **Keil µVision IDE**
  → Used for writing, compiling, and debugging the Embedded C code for the 8051 microcontroller.
- ❖ **Proteus Design Suite**
  → Used for simulating the microcontroller circuit, including switches, logic inputs, and virtual terminal display.
- ❖ **8051 Microcontroller (AT89C51)**
  → Core controller used in the project, simulated inside Proteus.
- ❖ **Virtual Terminal**
  → Used to display the elevator's current floor or alert messages in the simulation.
- ❖ **Logic State Inputs and Switches**
  → Used to simulate user input for floor selection and fire alerts.

2.1.3 Implementation Process

1. **Circuit Design**:
   A schematic was designed in Proteus where:

   - Push-button switches were connected to input pins for floor selection.
   - A logic input was connected to simulate the fire condition.
   - A virtual terminal was interfaced via UART to the 8051 to display outputs.

2. **Code Development**:
   Using Keil, the elevator control logic was written in Embedded C. The code:

   - Read floor input signals.
   - Displays the current floor.
   - Monitors fire alert logic.
   - Overrides elevator movement during emergencies.

3. **Compilation and Hex File Generation**:
   The code was compiled, and a `.hex` file was generated for simulation in Proteus.

4. **Microcontroller Programming in Proteus**:
   The compiled hex file was loaded into the simulated 8051 IC in Proteus.

5. **Testing and Debugging**:
   The simulation was run and tested for different input combinations to verify:

   - Proper floor transitions.
   - Correct emergency response on fire alert.
   - UART output messages for user awareness.

2.1.4 Simulation Process

- **Normal Operation**:
  When a floor button is pressed, the system updates the virtual terminal to indicate the elevator is moving to the selected floor.

- **Fire Alert Condition**:
  When the logic state simulating a fire is activated, the system halts all movements, overrides any current operation, and displays a fire alert message on the terminal.

❖ **Result Verification**:
Each operation was validated through Proteus waveform and terminal output observation, confirming the correctness of logic and expected system behavior.

2.2 Workflow of the system

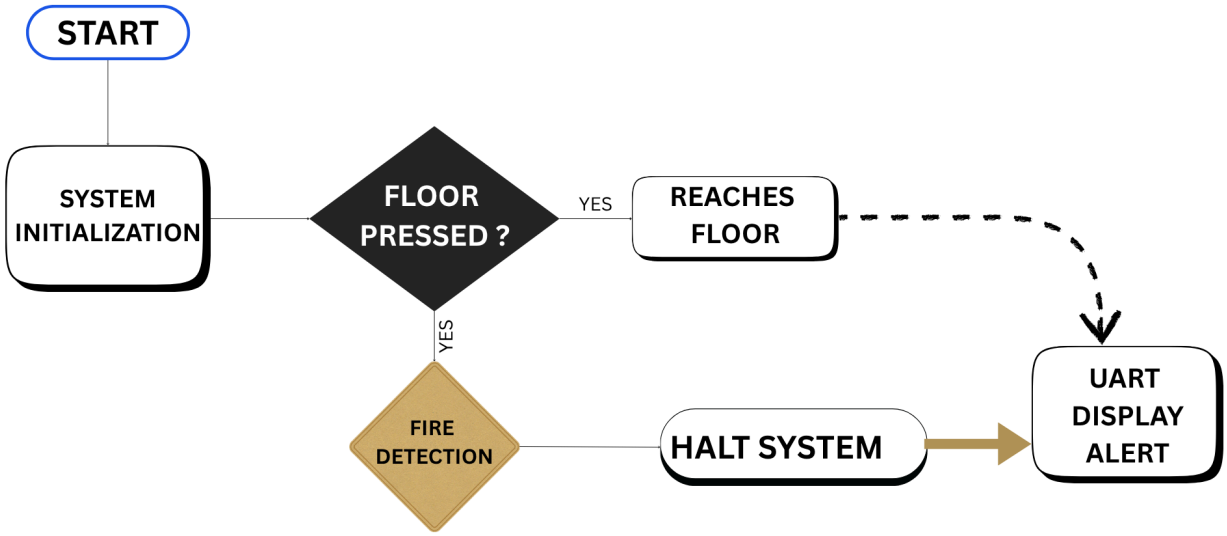


Fig 2.2 -Workflow of the system

2.2.1 Working Principle:

The working principle is based on **sensor-driven interrupt logic and motor control sequencing** using the AT89C51 microcontroller. The system uses **digital input detection** to identify floor selection or emergency conditions:

- **Floor Movement Logic:**
When a user presses a floor button, the corresponding logic level is read on Port 0. The system calculates the required number of steps and energizes the motor coils in sequence using outputs from Port 2 (P2.4–P2.7), effectively moving the elevator to the desired

floor.

- **Fire Detection:**
  The MQ2 sensor outputs a LOW signal on detection of gas/smoke. This triggers the fire_detected() function, which brings the elevator to ground floor and activates a buzzer connected to Port 1.5.

- **Tamper Alert:**
  The reed switch detects unauthorized access (door opened forcefully). When its logic goes LOW, the system halts the elevator and activates the buzzer for alert.

- **UART Messaging:**
  UART (via TXD on P3.1) sends textual feedback such as "Current Floor: 2" or "FIRE ALERT" to a Virtual Terminal for monitoring.

Thus, the entire system operates through a combination of **sequential motor control**, **sensor logic**, and **serial communication**.

2.2.2 System Requirements

Hardware Requirement

- Windows 10
- 8 GB RAM

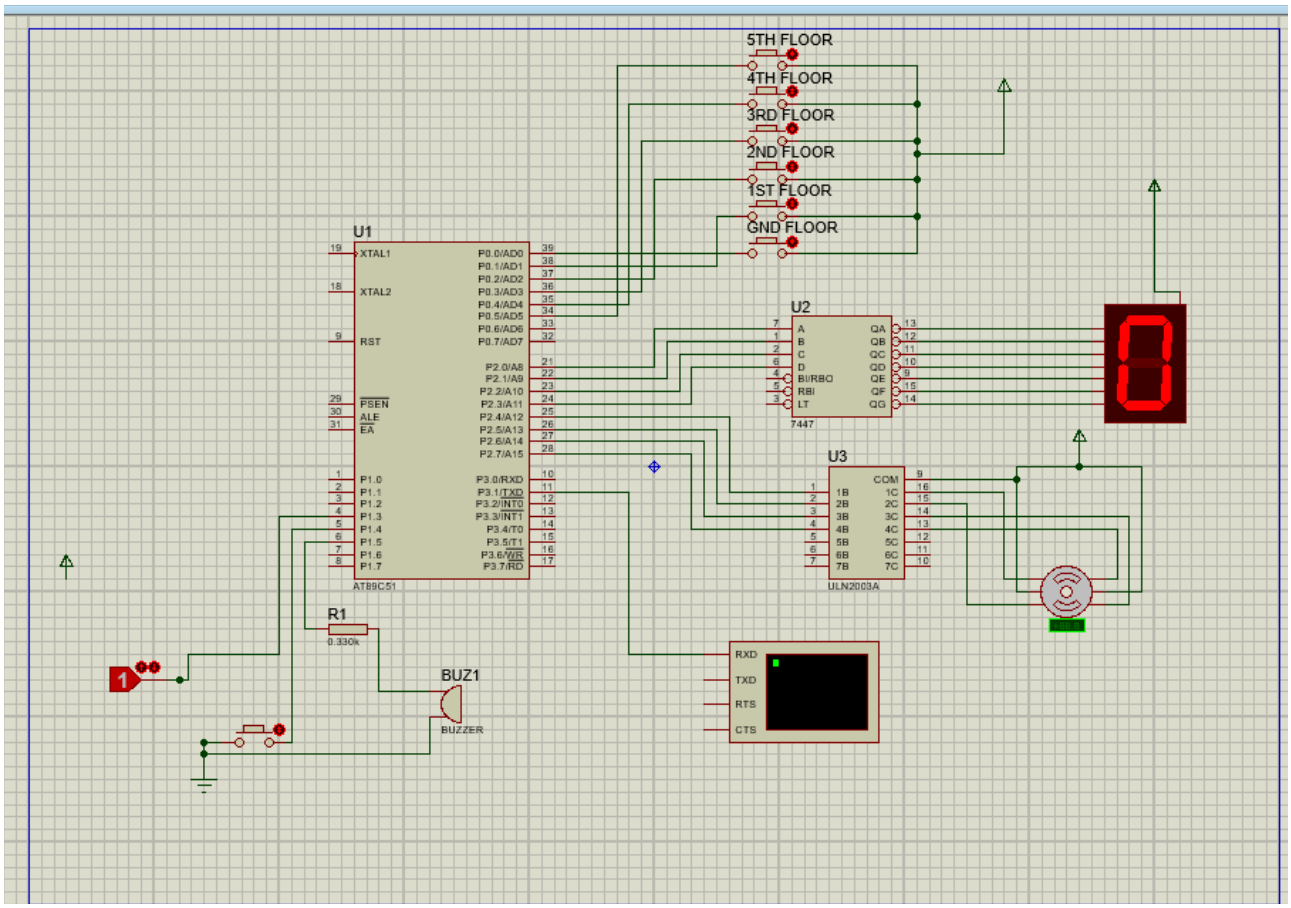
Software Requirement

- Proteus
- Keil u vision

Description Language

- Embedded C

Circuit Diagram:



CHAPTER 3 IMPLEMENTATION

3.1 Hardware Implementation (Proteus Simulation)

The system is implemented in Proteus Design Suite 8. The key components used and their virtual connections are as follows:

Component	Connection	Purpose
AT89C51 MCU	Central controller	Executes logic and controls peripherals
ULN2003A	P2.4–P2.7 to IN1–IN4	Drives stepper motor
Stepper Motor	ULN2003A OUT1–OUT4	Simulates elevator movement
Pushbuttons	P0.0–P0.5	Floor selection (0 to 5)
LOGIC STATE	DOUT to P1.3	Fire detection (active LOW)
Reed Switch	One end to P1.4, other to GND	Tamper detection
Buzzer	P1.5 through 330Ω resistor to GND	Audible alert in emergencies
Virtual Terminal	TXD (P3.1)	Displays UART messages (status & alerts)

All digital sensors are configured in Proteus to output either HIGH (no event) or LOW (alert condition).

The buzzer is activated by setting P1.5 HIGH when an alert is triggered.

3.2 Software Implementation (Keil µVision)

The software was developed in embedded C using Keil µVision IDE. Key code modules include:

3.2.1 Floor Control Logic

- The elevator floors (0 to 5) are selected via **push buttons connected to Port 0** (`P0.0` to `P0.5`).
- On detecting a valid input, the system:
    - Compare the **target floor** with the **current floor**.
    - Calls the `control()` function, which moves the elevator either up or down depending on the floor difference.

```
if (P0 == 0x01) { control(0); }

if (P0 == 0x02) { control(1); }

if (P0 == 0x04) { control(2); }

if (P0 == 0x08) { control(3); }

if (P0 == 0x10) { control(4); }

if (P0 == 0x20) { control(5); }
```

3.2.2 Stepper Motor Driver

A stepper motor is controlled via a ULN2003A driver, connected to P2.4 to P2.7.

Energizing sequences are executed through bit masking:

Clockwise Rotation (Elevator Going Up)

```
// Clockwise Stepper Sequence using P2.4–P2.7

P2 = (P2 & 0x0F) | 0x10;  // Step 1: 0001 xxxx → P2.4 HIGH

delay();

P2 = (P2 & 0x0F) | 0x20;  // Step 2: 0010 xxxx → P2.5 HIGH
```

```
delay();
P2 = (P2 & 0x0F) | 0x40;  // Step 3: 0100 xxxx → P2.6 HIGH
delay();
P2 = (P2 & 0x0F) | 0x80;  // Step 4: 1000 xxxx → P2.7 HIGH
delay();
```

- The `delay()` function is used to control stepping speed and ensure smooth transitions.

Counter-Clockwise Rotation (Elevator Going Down)

```
// Counter-Clockwise Stepper Sequence using P2.4–P2.7
P2 = (P2 & 0x0F) | 0x80;  // Step 1: 1000 xxxx → P2.7 HIGH
delay();
P2 = (P2 & 0x0F) | 0x40;  // Step 2: 0100 xxxx → P2.6 HIGH
delay();
P2 = (P2 & 0x0F) | 0x20;  // Step 3: 0010 xxxx → P2.5 HIGH
delay();
P2 = (P2 & 0x0F) | 0x10;  // Step 4: 0001 xxxx → P2.4 HIGH
delay();
```

3.2.3 Emergency Handling

1. Fire Detection (`fire_detected()`):

- Continuously monitors `P1.3` connected to the **Logic State OUT pin**.
- On detecting **logic LOW**, it:
    - Activates the **buzzer** via `P1.5`.
    - Send a `"FIRE ALERT"` message through UART.
    - Calls `control(0)` to return the elevator to the ground floor.

- Stop the system with `while(1);`.

2. Tamper Detection (`tamper_detected()`):

- Monitors `P1.4` connected to a **reed switch**.
- On detecting tampering (logic LOW):
  - Activates buzzer.
  - Send `"TAMPER ALERT"` through UART.
  - Brings elevator to ground floor.
  - Freezes further operations for safety.

```
if (FIRE == 0) {
    buzzer_on();
    uart_send_string("FIRE ALERT\r\n");
    control(0);
    buzzer_off();
    while (1);
}
```

3.2.4 UART Communication

- UART is initialized using Timer 1 in Mode 2:

```
void uart_init() {
    TMOD = 0x20;   // Timer1 in mode 2
    TH1 = 0xFD;    // Baud rate 9600
    SCON = 0x50;   // Serial mode 1: 8-bit data
    TR1 = 1;       // Start Timer1
    TI = 1;        // Enable transmit flag
```

```
}
```

- UART is used to send messages to the **Virtual Terminal** in Proteus for monitoring system status:
    - Current floor updates (e.g., `"SECOND FLOOR\r\n"`)
    - Emergency messages (`"FIRE ALERT\r\n"`, `"TAMPER ALERT\r\n"`)
- The `uart_send_string()` function sends strings byte-by-byte.

3.3 Integration and Testing

Once hardware and software were completed, the entire setup was tested in Proteus:

- ➢ **Normal operation**: Elevator moves correctly to selected floor.
- ➢ **Fire condition**: Logic state output LOW triggered ground floor return and buzzer.
- ➢ **Tamper condition**: Open reed switch halted system and activated alert.
- ➢ **UART output**: Real-time messages displayed in Virtual Terminal.

3.4 Simulation Process

The simulation process was carried out using **Proteus Design Suite 8 Professional** to test the functionality of the smart elevator system virtually, without using physical hardware. The simulation validated motor control, emergency detection, and UART communication.

3.4.1 Circuit Design in Proteus:

- The entire schematic was designed in Proteus.

3.4.2 HEX File Upload:

- The compiled .hex file from Keil µVision was uploaded to the AT89C51 microcontroller in Proteus by double-clicking the MCU and selecting the hex file under "Program File".

3.4.3 Running the Simulation:

- ❖ The simulation was started by clicking the Play (▶) button in Proteus.
- ❖ Floor buttons were clicked to simulate elevator calls.
- ❖ The elevator moved accordingly, and messages were displayed in the Virtual Terminal (e.g., "Current Floor: 2").

- When fire was simulated (LOGIC STATE = 0), the elevator automatically returned to the ground floor and the buzzer was activated.
- Similarly, tamper events triggered alerts and halted the elevator.
- The buzzer responded with logic HIGH when any alert occurred.

CHAPTER 4 RESULT

4.1 Simulation Output

After creating the proteus project with these components , run the simulation by uploading the hex file.

4.1.1 Moving to First Floor

When the user presses the button for the first floor, the system compares the current floor with the target, calculates the required number of steps, and energizes the stepper motor coils in a clockwise sequence via Port 2 (P2.4–P2.7) to move the elevator upward until it reaches the first floor.

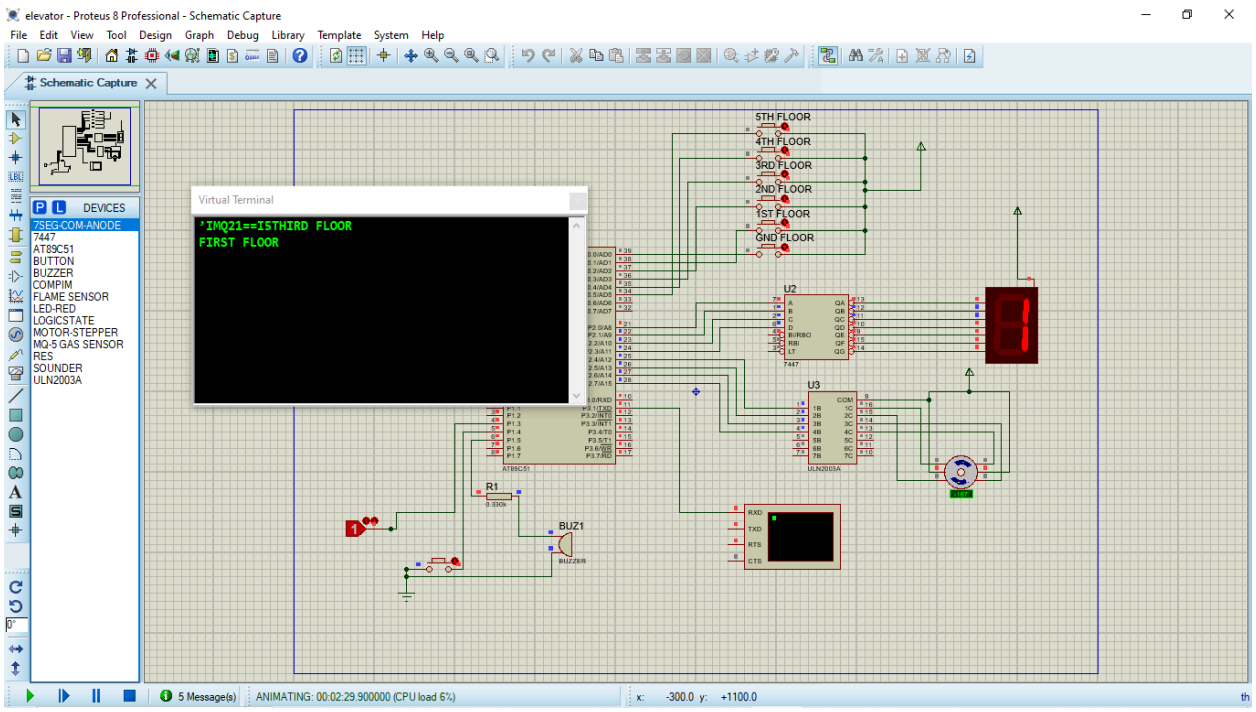


Fig 4.1.1 : FIRST FLOOR

4.1.2 Moving to Second Floor

Upon selection of the second floor, the system initiates a motor sequence similar to the first floor, rotating the stepper motor clockwise until the elevator reaches the second level, as tracked in the program.

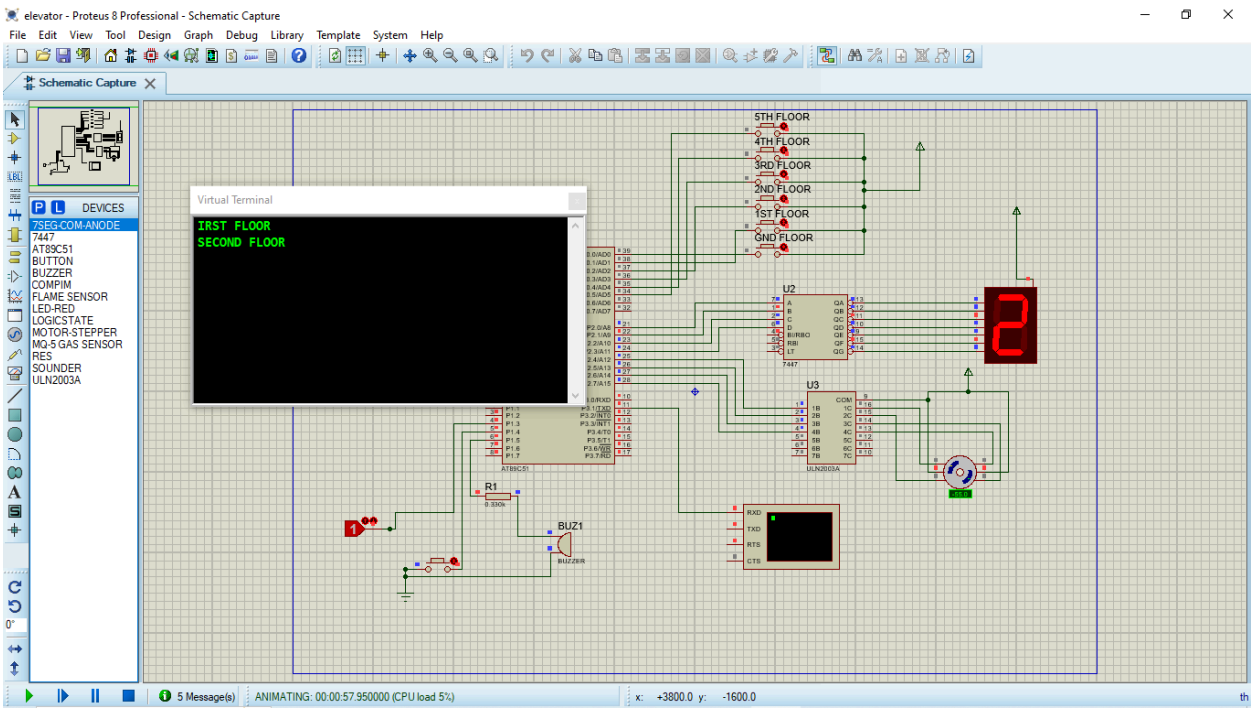


Fig 4.1.2: SECOND FLOOR

4.1.3 Moving to Third Floor

When the third-floor button is activated, the control logic computes the difference and drives the motor to move three levels from the ground. UART also logs the current floor during motion.

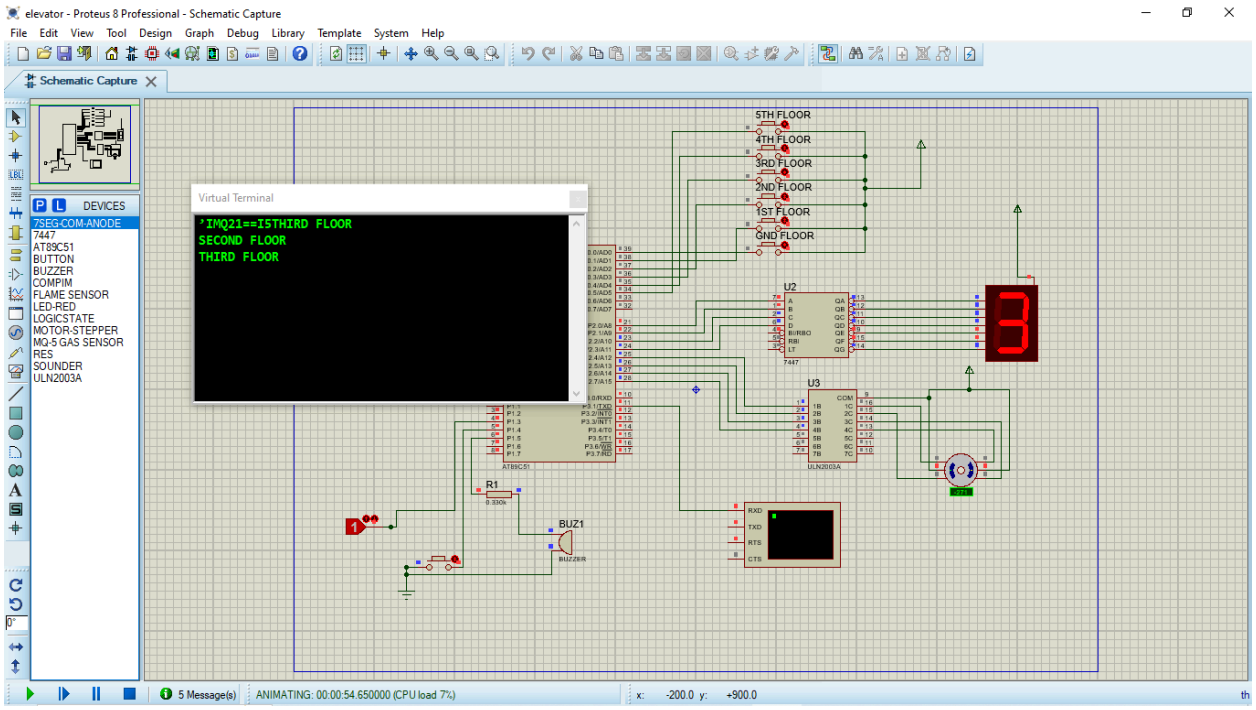


Fig 4.1.3: THIRD FLOOR

4.1.4 Moving to Fourth Floor

The elevator advances to the fourth floor by stepping through each level with timed delays between motor pulses. The system ensures that the elevator stops precisely at the fourth floor.

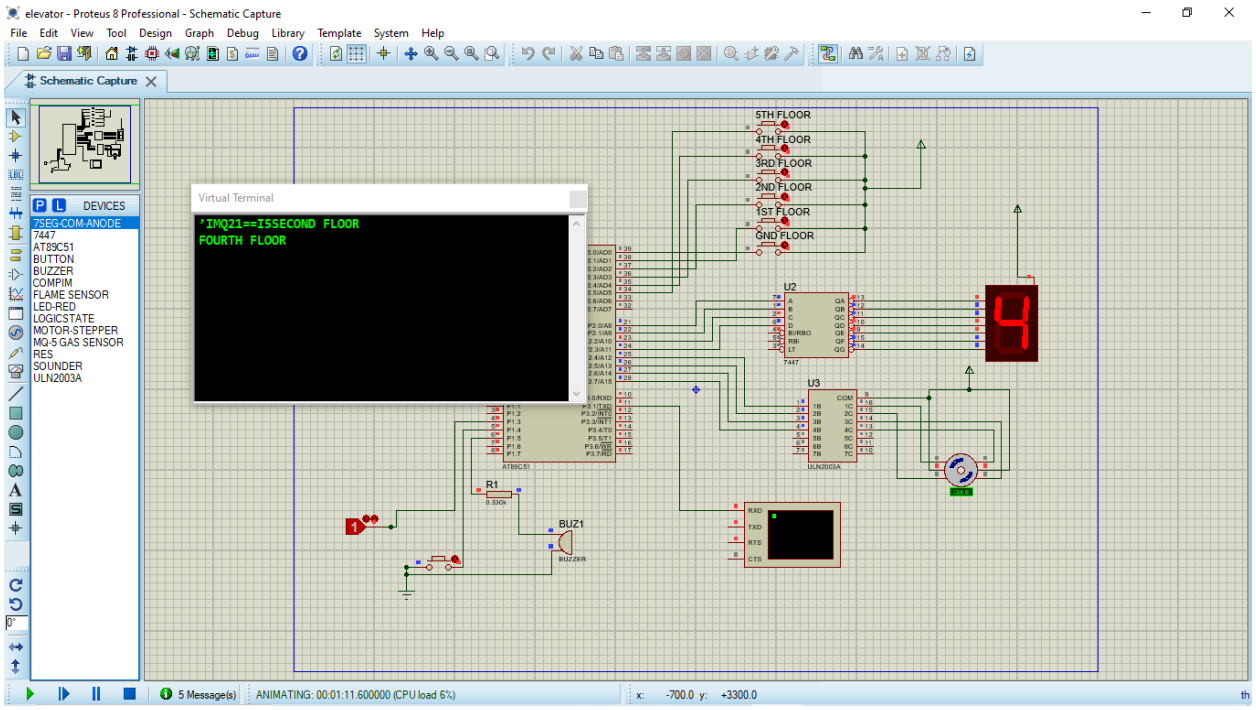


Fig 4.1.4: FOURTH FLOOR

4.1.5 Fire Alert Condition

When the LOGIC STATE goes LOW, the system triggers a fire alert. It immediately activates the buzzer, sends a "FIRE ALERT" message via UART, and returns the elevator to the ground floor.

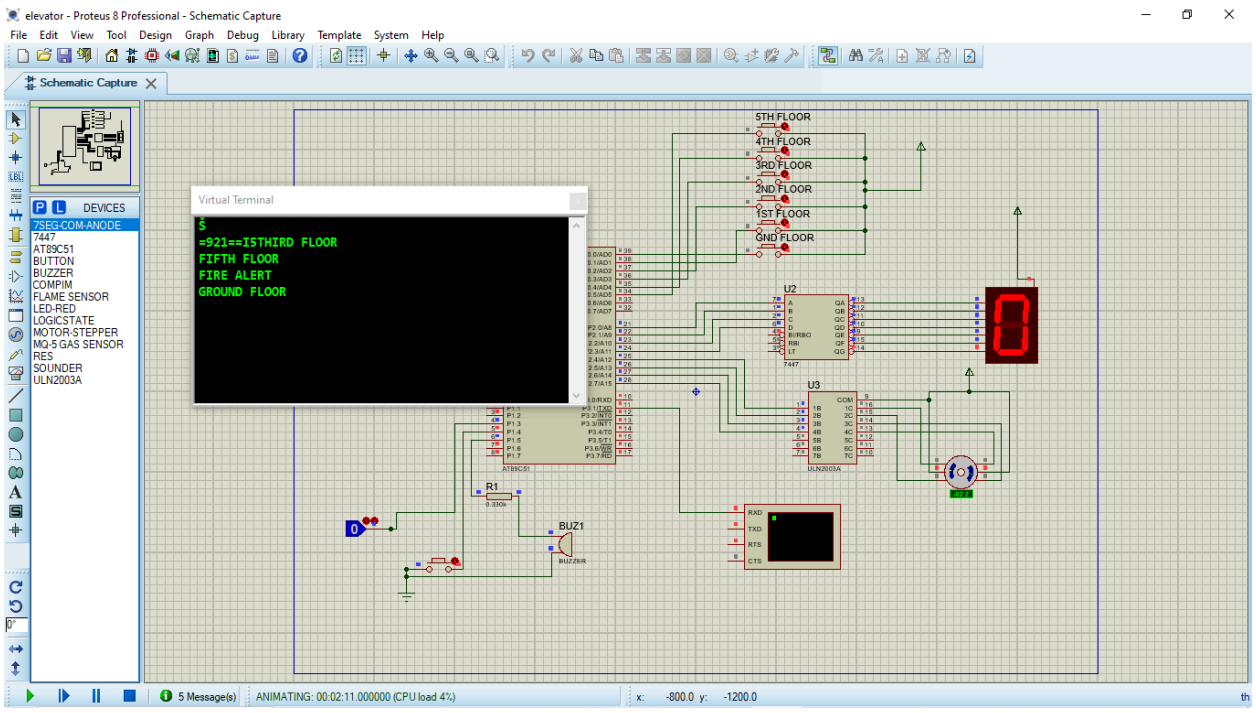


Fig 4.1.5:FIRE ALERT CONDITION

4.1.6 Tamper Alert Condition

If the reed switch detects unauthorized access or tampering (logic LOW), the system halts elevator movement, sounds an alert via buzzer, and prints a tamper message on the UART terminal.

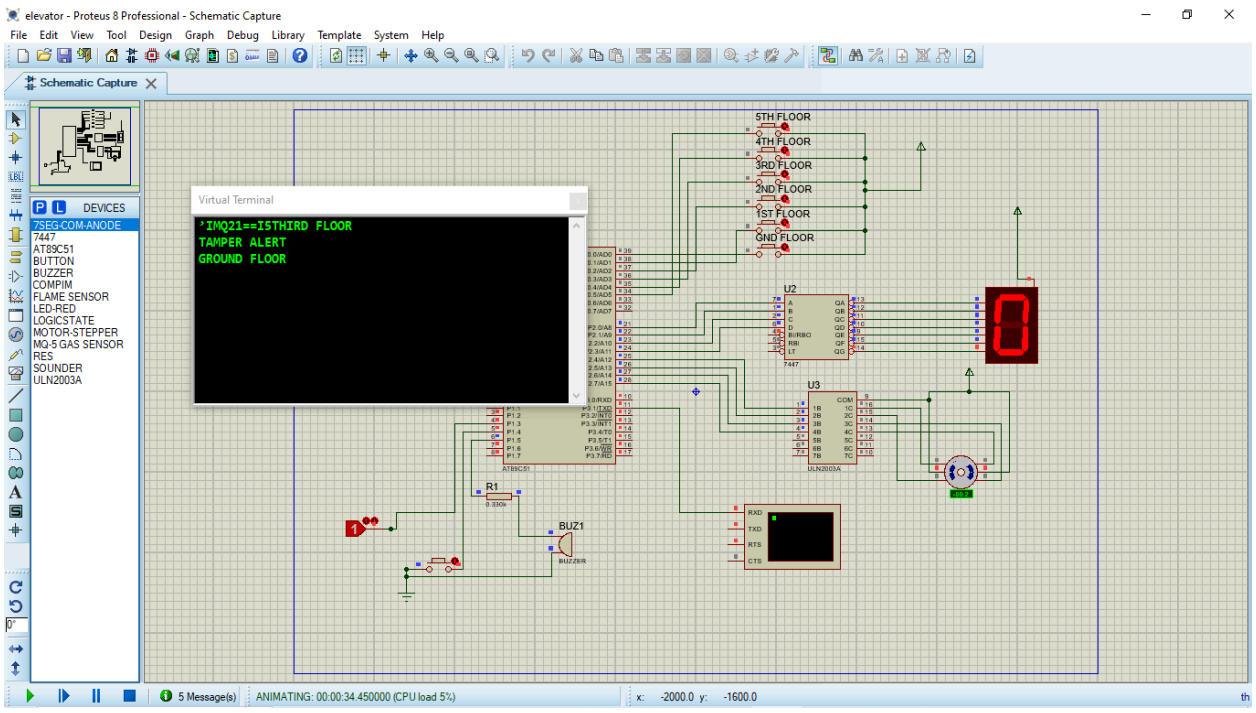


Fig 4.1.6:TAMPER ALERT CONDITION

4.1.7 All Conditions

The image illustrates elevator movement across multiple floors based on user input through push buttons. It also shows emergency scenarios—fire detection via Logic State and tamper alert through the reed switch—triggering the buzzer, UART messages, and immediate return to the ground floor for safety.

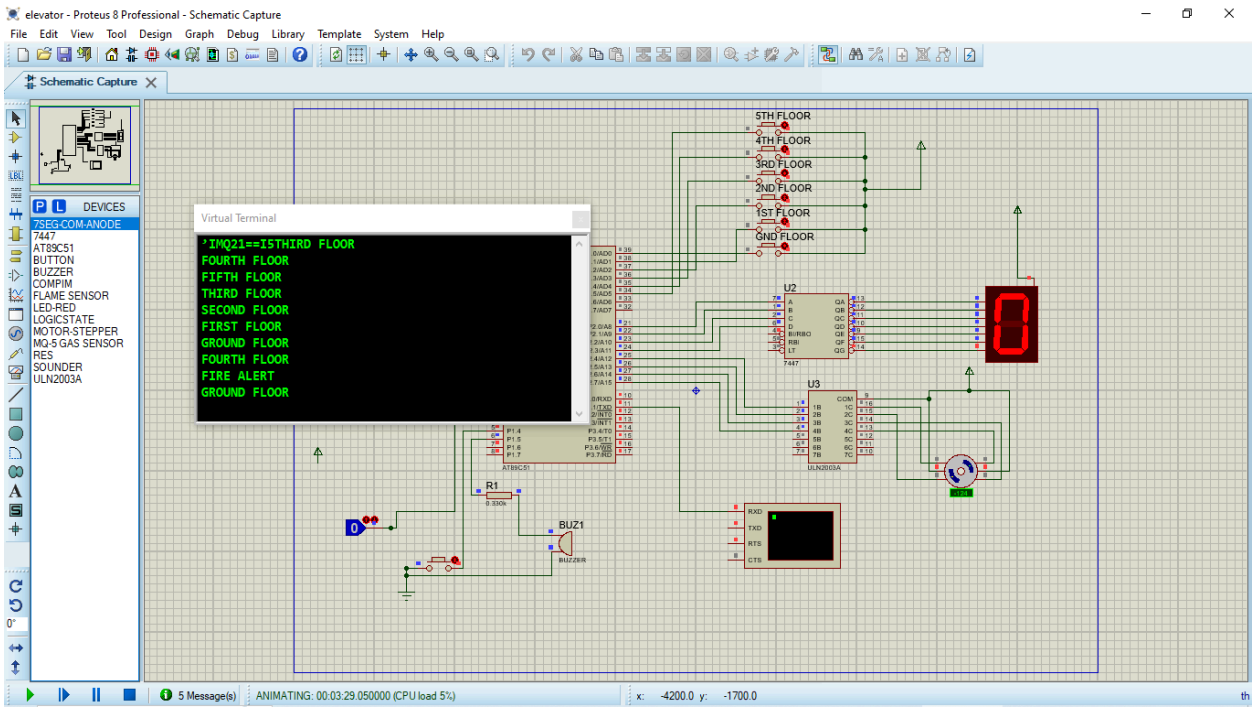


Fig 4.1.7: ALL CONDITIONS

CHAPTER 5 CONCLUSION

5.1.Summary

The Advanced and Safe Elevator System was successfully designed and simulated using the 8051 microcontroller, Keil µVision for programming, and Proteus for simulation. The core functionality of the elevator—floor movement control—was implemented using stepper motor logic through the ULN2003A driver, allowing smooth transitions between floors from ground to fifth. Each floor was individually selectable via dedicated push buttons, and motor control was handled with precise timing to simulate realistic movement.

Beyond basic movement, the system was enhanced with **safety and emergency handling features**. Two critical conditions—**fire detection** and **tampering**—were simulated using logic state inputs to represent sensors. When activated, these conditions triggered the buzzer and halted the system after safely returning the elevator to the ground floor. Additionally, **UART communication** provided real-time status updates and alerts through a virtual terminal, allowing external monitoring of the system's state and floor transitions. This brought in an extra layer of reliability and observability.

Overall, the project met its objective of not just simulating a functional elevator, but also integrating basic safety intelligence into the system. The use of modular code, proper port mapping, and real-time simulation tools made the development process structured and effective. This project has laid a strong foundation for understanding embedded control systems, motor interfacing, and emergency response logic—skills essential for designing smart, real-world automation systems.

5.2.References

1. Muhammad Ali Mazidi, Janice Gillispie Mazidi, and Rolin D. McKinlay,
   The 8051 Microcontroller and Embedded Systems: Using Assembly and C, 2nd Edition, Pearson Education, 2006.
2. Raj Kamal,
   Embedded Systems: Architecture, Programming and Design, 2nd Edition, McGraw Hill Education, 2011.
3. Keil µVision IDE – Official Documentation
   https://www.keil.com/support/man/docs/uv4
4. Labcenter Electronics,
   Proteus Design Suite – Official User Guide
   https://www.labcenter.com
5. Online Forums and Community Examples from:
   https://www.electronicwings.com
   https://www.allaboutcircuits.com
6. Datasheets for 8051 Microcontroller, ULN2003A Driver, and logic state tool in Proteus.