REVA UNIVERSITY
Bengaluru, India

**SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING**

MINI PROJECT REPORT
ON

# "Elevator Control System (2-3 Floors)"

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

**Bachelor of Technology**
**In**
**Electrical and Electronics Engineering**

Submitted by

HEMANTH H K (R22EM032)
JILAKARA ABHINAY (R22EM033)
GANGAMBIKA B K (R22EM026)

TO

**Prof. Gopinath Sir**
School of EEE
REVA UNIVERSITY

**2024-2025**

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru-560064
www.reva.edu.in

# 2–3 Floor Elevator Control System using ESP32 and MicroPython

## Table of Contents

## 1. Abstract

This project presents the design, development, and testing of a logic-controlled elevator system operating across two to three floors. The controller is based on the ESP32 microcontroller, programmed using MicroPython—a lightweight scripting language tailored for embedded applications. The elevator operates by responding to button inputs to select the desired floor and moves the elevator using a DC motor driven by an L298N motor driver. Limit switches are used to detect floor arrivals and stop the motor precisely at each level. This project combines hardware control and software logic to create a working prototype of a simplified elevator, with potential applications in educational and prototyping environments. The system's modular design allows for expansion to more floors or integration with advanced features like displays or IoT controls

## 2. Introduction

Elevator systems are an integral part of modern infrastructure, from homes to skyscrapers. Understanding their working mechanism offers insight into automation, embedded control, and electromechanical integration. This project aims to replicate the logic and functionality of a basic elevator system for 2–3 floors, focusing on how user inputs (button presses) are translated into mechanical actions using microcontrollers and actuators. By using an ESP32—a powerful microcontroller with Wi-Fi and Bluetooth support—we open opportunities for future enhancements like wireless control or telemetry. Programming is done using MicroPython for its ease of use, cross-platform compatibility, and suitability for rapid development. The aim is to not just simulate, but to physically build and test the elevator’s operations in real time.

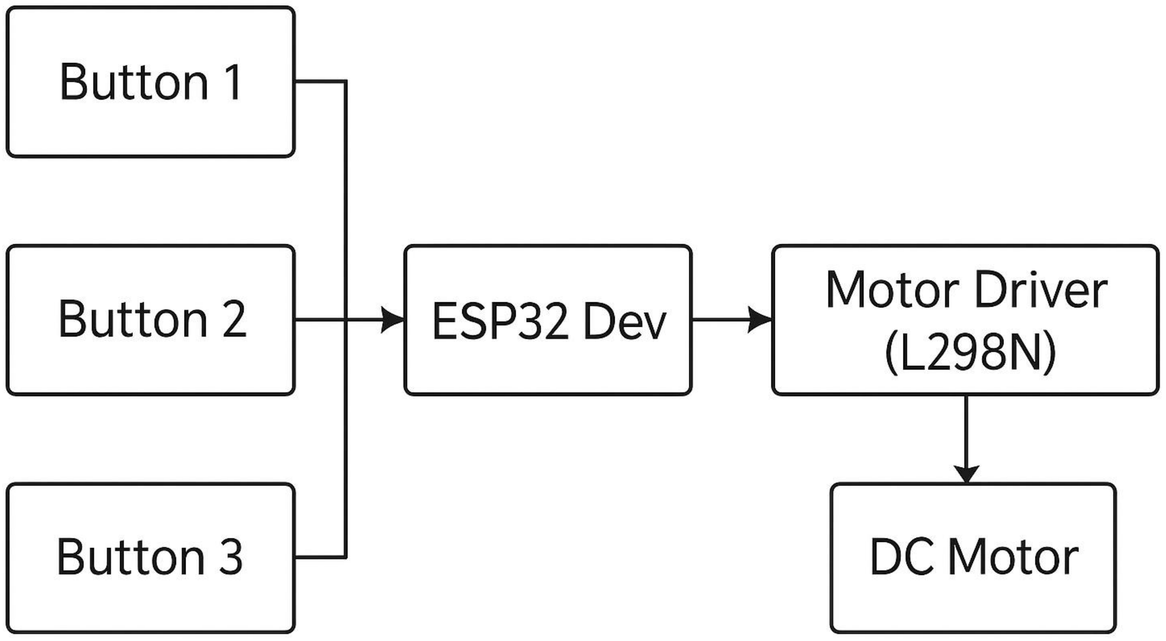
## 3. Objectives

- To simulate the functionality of a basic elevator serving 2–3 floors.
- To apply MicroPython programming to control physical hardware.
- To demonstrate GPIO input/output management in a real-world application.
- To teach core embedded systems concepts like polling, debouncing, motor control, and sensor interfacing.
- To design a scalable system that can be expanded beyond 3 floors or integrated with IoT features.

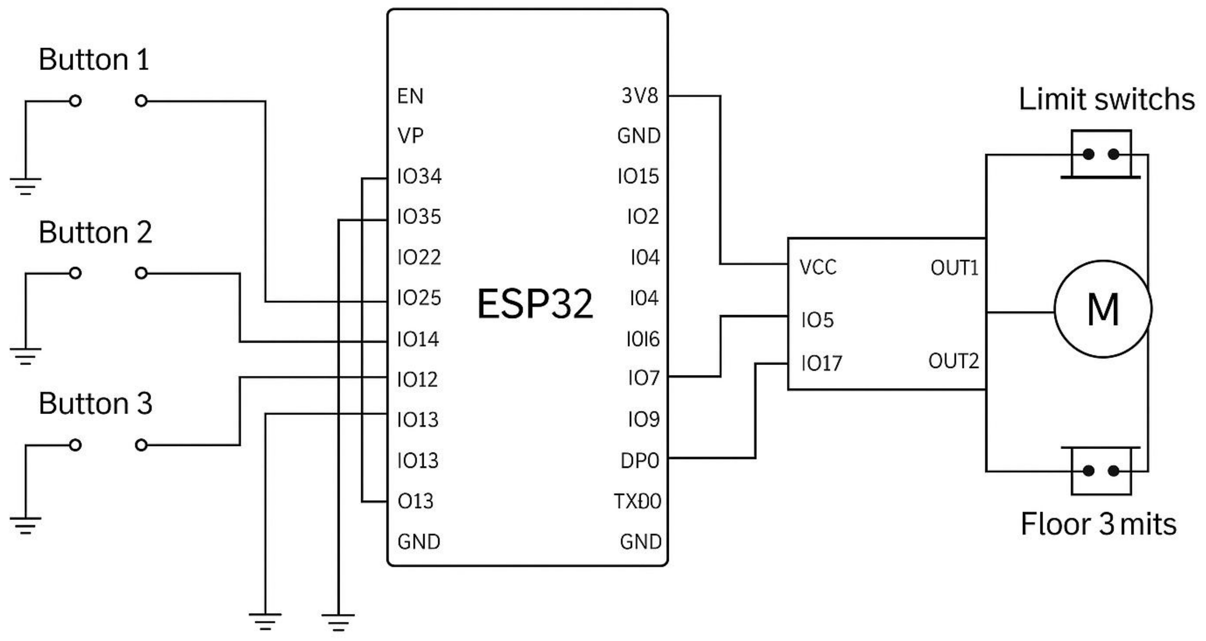
## 4. Components Required

| COMPONENT | QUANTITY |
|---|---|
| | |
| ESP32 DEV BOARD | 1 |
| L298N MOTOR DRIVER | 1 |
| DC MOTOR | 1 |
| PUSH BUTTONS | 3 |
| LIMIT SWITCHES | 3 |
| JUMPER WIRES | As needed |
| BREADBOARD | 1 |
| 12V POWER SUPPLY | 1 |

## 5. Block Diagram



## 6. Circuit Diagram



2−3 Floor Elevator Control System

### 6.1. Circuit Diagram Explanation

The circuit is designed with:

- **Three push buttons**, each wired with internal pull-ups on ESP32, connected to GPIO22, GPIO25, and GPIO14. Pressing the button connects the input to ground, registering as a LOW signal.

- **Three limit switches** are used to detect floor arrival. They're also connected using pull-ups to GPIO33, GPIO32, and GPIO35.

- The **L298N motor driver** has two input pins connected to GPIO5 and GPIO17, allowing bidirectional motor control.

- The **DC motor** is powered separately from a 12V power source, ensuring enough torque.

## 7. Working Principle

1. The system is initialized in idle mode.
2. When a user presses a button, the ESP32 identifies the requested floor.
3. The motor is activated via the L298N driver to move in the direction (up/down) towards that floor.
4. Once the respective floor's **limit switch** is pressed (indicating elevator arrival), the ESP32 stops the motor.
5. The elevator remains at that floor until another floor is requested.

## 8. ESP32 Pin Configuration

| **Function** | **ESP32 Pin** |
|---|---|
| Button 1 (Floor 1) | GPIO22 |
| Button 2 (Floor 2) | GPIO25 |
| Button 3 (Floor 3) | GPIO14 |
| Motor IN1 | GPIO5 |
| Motor IN2 | GPIO17 |
| Floor 1 Switch | GPIO33 |
| Floor 2 Switch | GPIO32 |
| Floor 3 Switch | GPIO35 |

## 9. MicroPython Code

```
from machine import Pin
import time
# Button pins
button1 = Pin(22, Pin.IN, Pin.PULL_UP)
button2 = Pin(25, Pin.IN, Pin.PULL_UP)
button3 = Pin(14, Pin.IN, Pin.PULL_UP)

# Motor driver pins
```

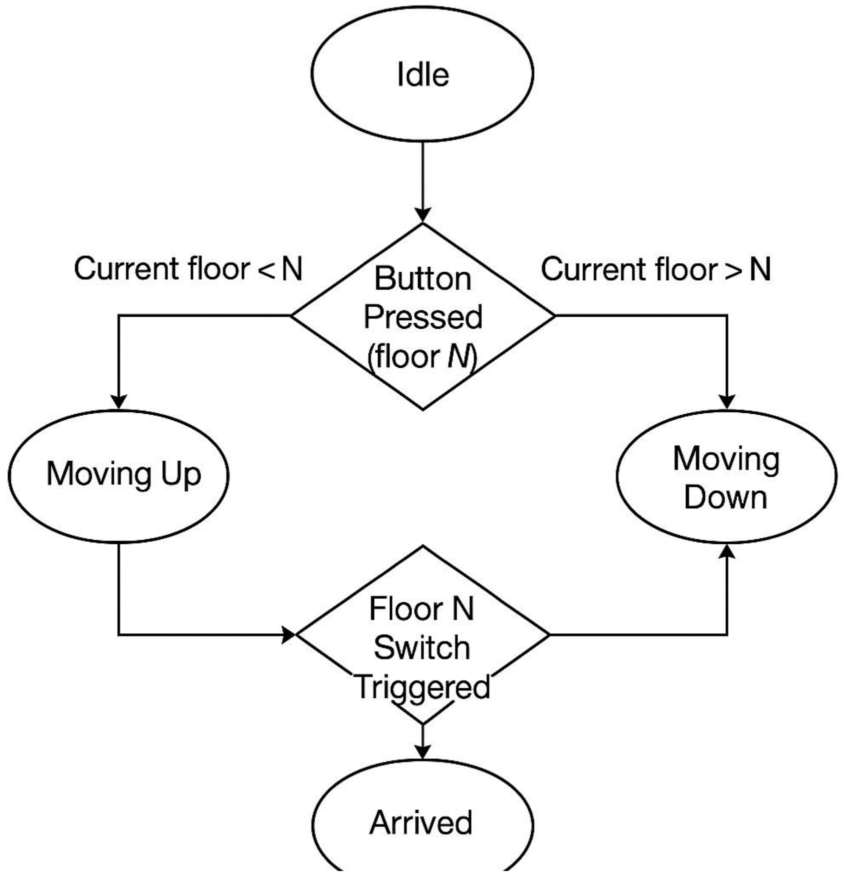
```
motor_in1 = Pin(5, Pin.OUT)
motor_in2 = Pin(17, Pin.OUT)

# Limit switches (floor sensors)
floor1_switch = Pin(33, Pin.IN, Pin.PULL_UP)
floor2_switch = Pin(32, Pin.IN, Pin.PULL_UP)
floor3_switch = Pin(35, Pin.IN, Pin.PULL_UP)

def move_up():
    motor_in1.on()
    motor_in2.off()

def move_down():
    motor_in1.off()
    motor_in2.on()

def stop_motor():
    motor_in1.off()
    motor_in2.off()

def go_to_floor(target_floor):
    while True:
        if target_floor == 1:
            move_down()
            if floor1_switch.value() == 0:
                stop_motor()
                break
        elif target_floor == 2:
            move_up()
            if floor2_switch.value() == 0:
                stop_motor()
                break
        elif target_floor == 3:
            move_up()
            if floor3_switch.value() == 0:
                stop_motor()
                break
        time.sleep(0.1)

while True:
    if button1.value() == 0:
        print("Button 1 pressed - Going to Floor 1")
        go_to_floor(1)
    elif button2.value() == 0:
        print("Button 2 pressed - Going to Floor 2")
        go_to_floor(2)
    elif button3.value() == 0:
        print("Button 3 pressed - Going to Floor 3")
        go_to_floor(3)
    time.sleep(0.1)
```

## 10. MicroPython Code Explaination

The code initializes input and output pins. When a button is pressed:

- The function go_to_floor() is called with the desired floor.
- Motor direction is decided based on the target floor.
- Continuous polling of the respective limit switch allows the elevator to stop precisely.

## 11. State Flow



## 12. Applications

- Home-based dumbwaiters
- Industrial material lifts
- Educational project for embedded and automation courses
- Scaled down prototype of real-world elevators

## 13. Advantages and Limitations

### Advantages

- Simple, scalable design
- Uses open-source toolsEasy to expand with IoT and sensors
- Low power consumption (ESP32)

### Limitations

- Limited to 3 floors unless expanded
- Lacks feedback like display/position tracking
- Basic safety mechanisms (e.g., no door simulation)

## 14. Future Scope

- Integration with **LCD or OLED** displays for floor status
- Use of **Bluetooth/Wi-Fi** for remote operation
- Add **buzzer**, **door lock simulation**, and **weight sensors**
- Use **stepper motors with encoders** for more precise control
- Upgrade to a **touch panel interface**

## 15. Conclusion

This project successfully models the logic of a basic elevator system using ESP32 and MicroPython. It demonstrates how embedded systems can be applied to real-world electromechanical control problems. By combining hardware and software logic, we have built a scalable and educational prototype that serves as a foundation for more complex systems.