Assignment 9

Name : Samyak Shah

Class : TE Comp

Roll : 9085

Title:

Writing an application using Raspberry-Pi board to control the operation of a hardware simulated Lift Elevator

Aim/Objectives:

- To understand the working principle of Lift Elevator
- To interface the Lift Elevator module with Raspberry Pi model
- To program the Raspberry Pi model to control operation of Lift Elevator module

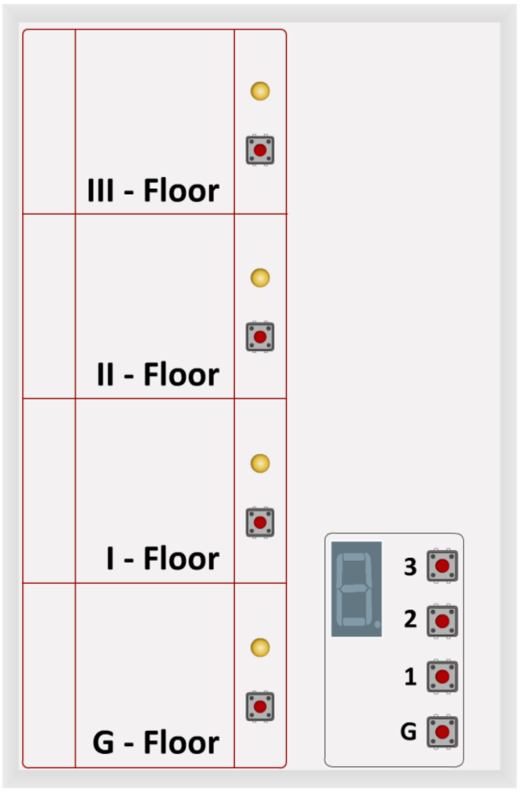
Software:

- Raspbian OS (IDLE)

Hardware Modules:

- Raspberry Pi Board module
- Lift Elevator module

Theory:

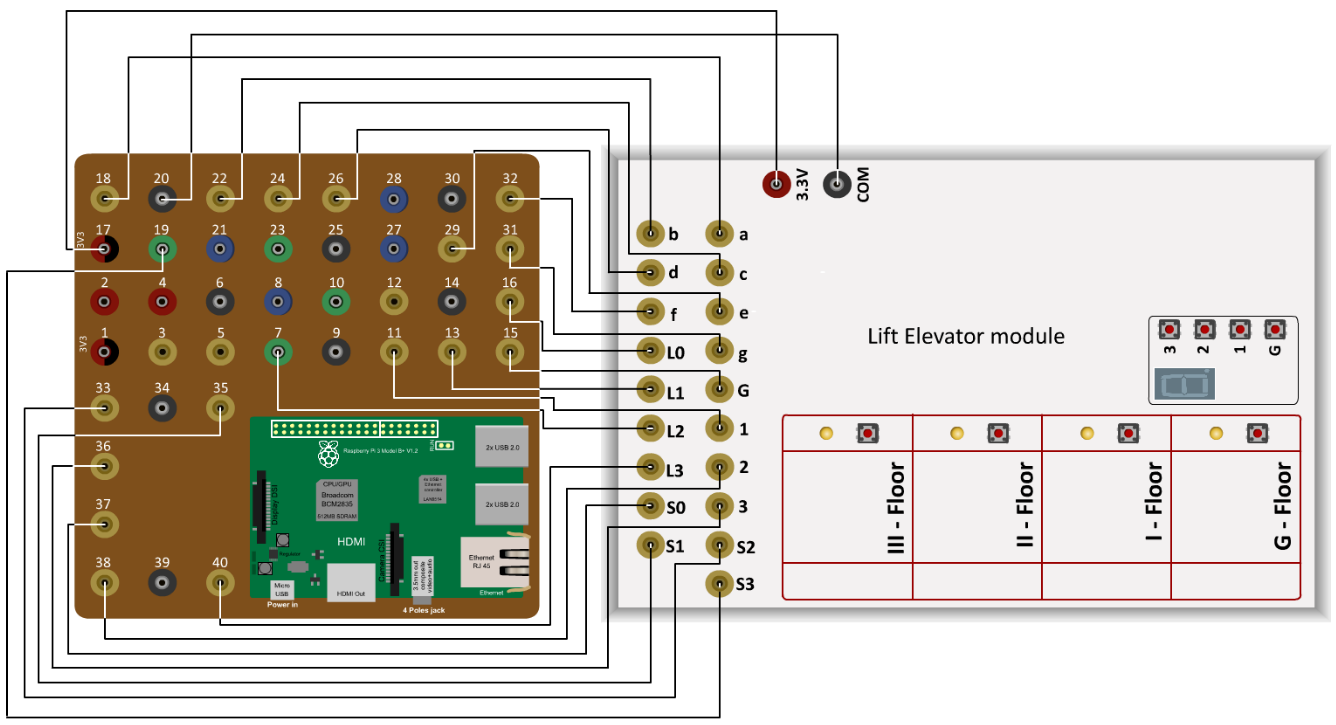


- Lift Elevator Module has two parts:
    a. Moving part inside the lift and
    b. Stationary part outside the lift at each floor to call the lift
- In this simulation module, we have considered four floors of a building
- So the Moving part contains four Push buttons. Out of these four buttons, one button is for each floor having floor number written below it. User has to push one of these buttons as the destination floor.
- The moving part also contains a Seven Segment Display to indicate the current floor number when the lift is moving.
- By pressing one of these buttons, the user indicates the destination floor.
- At each floor, the stationary part contains a buttons for calling the lift.
- In real life, when the lift is called by any floor, the lift starts moving towards the particular floor. When it reaches the particular floor, the lift door is opened.
- In our module, this situation is indicated by the “LED ON” status. So the LED ON status indicates that the lift has arrived at the particular floor, its door is opened and the user is entering the lift.
- In real life, as soon as the entering users get finished, the lift door is closed and the lift starts moving toward the destination.
- In our module, this situation is indicated by “LED OFF” status. So the LED OFF status indicates that the lift is moving towards the destination floor.

Safety precautions:

- Raspberry-Pi provides 3.3V and 5V VCC pins □ Raspberry-Pi operates on 3.3V.
- Various sensors and actuators operate on different voltages.
- Read datasheet of a given sensor or an actuator and then use appropriate VCC pin to connect a sensor or an actuator.
- Ensure that signal voltage coming to the Raspberry-Pi from any sensor or actuator does not exceed 3.3V.
- If signal/data coming to Raspberry-Pi is greater than 3.3V then use voltage level shifter module to decrease the incoming voltage.
- The Raspberry-Pi is a costly device, hence you should show the circuit connections to your instructor before starting your experiment.

Interface diagram:



Steps for assembling circuit:

- Connect all the pins of Lift Elevator module to pins of Raspberry Pi module as shown in the above figure.

Procedure:

- Write the program as per the algorithm given.
- Save the program
- Run code using Run module.

Algorithm:

- Import GPIO and time libraries
- Set GPIO mode as per Board
- Declare four Push button pins of the stationary part (outside the lift at each floor for calling the lift).
- Declare four LED pins at each floor for detection of door close and open.
- Declare four Push button pins of the moving part (inside the lift for selecting the destination)
- Declare seven pins of Seven Segment Display (this indicates the current floor number of the moving lift)
- Set the Push button pins as Input.
- Set the seven segment display pins and LED pins as Output.
- Store the value of each digit of seven segment display in variables.
- In the while loop, If “Floorbutton0” is pressed then lift at Ground floor and LED at Ground floor get ON for 5 second then gets OFF (door close). This step is repeated for Floorbutton1, Floorbutton2, Floorbutton3.
- Person enters in the lift and presses the push button of any one floor in the moving lift.
- The Seven Segment Display displays the floor number of the destination.

Observation:

- Observe the output on LEDs and Seven Segment Display.

Code:

```
import RPi.GPIO as GPIO
import time
import signal
import sys

HIGH=1
LOW=0
#------------------ Rpi Config
RUNNING= True
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
#var for disp  // out
a=18
b=22
c=24
d=26
e=29
f=32
g=31
# lift floor pos //out
l0=16
l1=13
l2=5
l3=40
#lift inside buttons (0,1,2,3) //in
i0=15
i1=11
i2=38
i3=36
# B floor buttons// out
b0=37
b1=35
b2=33
b3=19


def init_all():
    GPIO.setup(a,GPIO.OUT)
    GPIO.setup(b,GPIO.OUT)
    GPIO.setup(c,GPIO.OUT)
    GPIO.setup(d,GPIO.OUT)
    GPIO.setup(e,GPIO.OUT)
    GPIO.setup(f,GPIO.OUT)
    GPIO.setup(g,GPIO.OUT)
    #-----------------------
    GPIO.setup(l0,GPIO.OUT)
    GPIO.setup(l1,GPIO.OUT)
    GPIO.setup(l2,GPIO.OUT)
    GPIO.setup(l3,GPIO.OUT)
    #-------------------------
    GPIO.setup(i0,GPIO.IN)
    GPIO.setup(i1,GPIO.IN)
    GPIO.setup(i2,GPIO.IN)
    GPIO.setup(i3,GPIO.IN)
    #--------------------------
    GPIO.setup(b0,GPIO.IN)
    GPIO.setup(b1,GPIO.IN)
    GPIO.setup(b2,GPIO.IN)
    GPIO.setup(b3,GPIO.IN)

#Display coding
def disp(f_n0):
    #
    if f_n0==0:
        GPIO.output(a, HIGH)
        GPIO.output(b, HIGH)
        GPIO.output(c, HIGH)
        GPIO.output(d, HIGH)
        GPIO.output(e, HIGH)
        GPIO.output(f, HIGH)
        GPIO.output(g, LOW)
    elif f_n0==1:
        GPIO.output(a, LOW)
        GPIO.output(b, HIGH)
        GPIO.output(c, HIGH)
        GPIO.output(d, LOW)
        GPIO.output(e, LOW)
        GPIO.output(f, LOW)
        GPIO.output(g, LOW)
    elif f_n0==2:
        GPIO.output(a, HIGH)
        GPIO.output(b, HIGH)
        GPIO.output(c, LOW)
        GPIO.output(d, HIGH)
        GPIO.output(e, HIGH)
        GPIO.output(f, LOW)
        GPIO.output(g, HIGH)
    elif f_n0==3:
        GPIO.output(a, HIGH)
        GPIO.output(b, HIGH)
        GPIO.output(c, HIGH)
        GPIO.output(d, HIGH)
        GPIO.output(e, LOW)
        GPIO.output(f, LOW)
        GPIO.output(g, HIGH)
    else:
        GPIO.output(a, LOW)
        GPIO.output(b, LOW)
        GPIO.output(c, LOW)
        GPIO.output(d, LOW)
        GPIO.output(e, LOW)
        GPIO.output(f, LOW)
        GPIO.output(g, LOW)



def lift_pos_led(f_n0):
    if f_n0==0:
        GPIO.output(l0, HIGH)
        GPIO.output(l1, LOW)
        GPIO.output(l2, LOW)
        GPIO.output(l3, LOW)
    elif f_n0==1:
        GPIO.output(l0, LOW)
        GPIO.output(l1, HIGH)
        GPIO.output(l2, LOW)
        GPIO.output(l3, LOW)
    elif f_n0==2:
        GPIO.output(l0, LOW)
        GPIO.output(l1, LOW)
        GPIO.output(l2, HIGH)
        GPIO.output(l3, LOW)
    elif f_n0==3:
        GPIO.output(l0, LOW)
        GPIO.output(l1, LOW)
        GPIO.output(l2, LOW)
        GPIO.output(l3, HIGH)

def lift_pos_led_off():
    print "low"
    GPIO.output(l0, LOW)
    GPIO.output(l1, LOW)
    GPIO.output(l2, LOW)
    GPIO.output(l3, LOW)



#-----------------


#


# main code Start
print("I AM INIT")
GPIO.cleanup()
init_all()
```

```
ctr=0
state0=0
state1=0
state2=0
state3=0
lift_pos_led_off()
GPIO.output(l0, HIGH)
print l3
disp(0)
lift_pos_led(0)
time.sleep(5);
'''
disp(0)
lift_pos_led(0)
time.sleep(2);
disp(1)
lift_pos_led(1)
time.sleep(2);
disp(2)
lift_pos_led(2)
time.sleep(2);
disp(3)
lift_pos_led(3)
'''
try:
    while True:
        time.sleep(1);
        state0=GPIO.input(i0)
        state1=GPIO.input(i1)
        state2=GPIO.input(i2)
        state3=GPIO.input(i3)
        time.sleep(1);
        print "i0"
        print state0
        print "i1"
        print state1
        print "i2"
        print state2
        print "i3"
        print state3
        if state0:
            while ctr>=0:
                ctr-=1
                time.sleep(1);
                #lift_pos_led_off()
                disp(ctr)
                lift_pos_led(ctr)
            lift_pos_led(0)
            disp(0)
            ctr=0
            lift_pos_led(ctr)
            state0=0
        elif state1:
```

```
    if ctr>1:
        while ctr>=1:
            ctr-=1
            time.sleep(1);
            #lift_pos_led_off()
            disp(ctr)
            lift_pos_led(ctr)
        lift_pos_led(1)
        disp(1)
        ctr=1
        lift_pos_led(ctr)
    else:
        while ctr<=1:
            ctr+=1
            time.sleep(1);
            #lift_pos_led_off()
            disp(ctr)
            lift_pos_led(ctr)
        lift_pos_led(1)
    disp(1)
    ctr=1
    lift_pos_led(ctr)
    state1=0
elif state2:
    if ctr>1:
        while ctr>=2:
            ctr-=1
            time.sleep(1);
            #lift_pos_led_off()
            disp(ctr)
            lift_pos_led(ctr)
        lift_pos_led(2)
        disp(2)
        ctr=2
        lift_pos_led(ctr)
    else:
        while ctr<=2:
            ctr+=1
            time.sleep(1);
           #lift_pos_led_off()
            disp(ctr)
            lift_pos_led(ctr)
        lift_pos_led(2)
    disp(2)
    ctr=2
    lift_pos_led(ctr)
    state2=0
elif state3:
    while ctr<=3:
            ctr+=1
            time.sleep(1);
            #lift_pos_led_off()
            disp(ctr)
```

```
                    lift_pos_led(ctr)
                lift_pos_led(3)
                disp(3)
                ctr=3
                lift_pos_led(ctr)
                state3=0

finally:
    # Stop and finish cleanly so the pins
    # are available to be used again
    GPIO.cleanup()
```