**Experiment No. 11**

**Title:** Write an application using Raspberry-Pi /Beagle board to control the operation of a hardware simulated lift elevator

**Aim:** To understand the interface between Lift elevator simulation device and Raspberry Pi Board.

Demonstrate a lift elevator working system using Raspberry Pi and Python.

**Hardware Requirement**:

Raspberry Pi board with an SD card and the usual peripherals, we will also  need:

* 1x Solderless breadboard
* All king of jumper leads
* 1x Tactile button
* LEDs
* Switches/button

or a **Simulated life elevator board.**

**Software Requirement:** Raspbian O.S, Python.

**Theory:**

**GPIO pins**

One powerfull feature of the Raspberry Pi is the row of GPIO pins along the top edge of the boadr. GPIO stands for General-Purpose Input/Output. These pins are a physical interface between the Raspberry Pi and the outside world.

**3.3 volts**

Anything connected to these pins will always get 3.3 v of power.

**5 volts**

Anything connected to these pins will always get 5 v of power.

**GND**

Zero volts, used to complete a circuit

**GPIO**

These pins are for general-purpose use and can be configures as input or output pins

**ID\_SC/ID\_SD/DNC**

Special purpose pins.

**Lighting an LED**

LEDs are delicate little things. if you put too much current through them they will pop. To limit the current going through the LED, you should always use a register in series with it.

Long leg an LED to the Pi 3.3v and the short leg to a GND pin. Automatically LED will turn ON

**Making Traffic Light Signal**

1. No of floors for Lift Simulation Operation = 4

2. No of LEDs used for the lift direction (on Board) = 7

3. The floor no. where lift is positioned when program is executed. = 0 (Ground floor)

|  |  |  |
| --- | --- | --- |
| LEDs | RaspberryPi Pin Nos used | Explanation |
| Position LED | FL0\_LED\_POS = 10  FL1\_LED\_POS = 12  FL2\_LED\_POS = 13  FL3\_LED\_POS = 16 | Position LED will remain HIGH where the lift is currently placed. |
| Call Position LED | FL0\_LED\_CALL = 6  FL1\_LED\_CALL = 7  FL2\_LED\_CALL = 8  FL3\_LED\_CALL = 9 | Call Position LED will remain HIGH when there is a call made from that floor. This is the destination floor where lift call is made. |
| Direction LED | LED\_D0 = 21  LED\_D1 = 22  LED\_D2 = 23  LED\_D3 = 24  LED\_D4 = 25  LED\_D5 = 26  LED\_D6 = 27 | Direction LEDs are becoming HIGH during the time when lift is moving from one floor to another. It will be in loop until Lift reaches to its destination called floor.  The directions of these LED is based upon the call either Bottom to up or Up to bottom. |
| Switches | FL0\_SW = 17  FL1\_SW = 18  FL2\_SW = 19  FL3\_SW = 20 | Switches are the button pressed from user in order to make a call to lift at some specific floor. |

All the above LEDs are set in GPIO.Output mode as LEDs are getting HIGH or LOW based on the elevator call and moving status.

Except Switches which are working as GPOIO.IN mode as it receives the input from the user when a call to elevator is made.

**Implementation:**

For implementing the lift elevator using Raspberry Pi, we first initialize the lift position such as ground floor. Now as soon as the call is made from different floor a condition is check to make sure wether lift has to go UP or DOWN. The Position LED is set HIGH for that floor. Now direction LED starts blinking from one direction (either bottom -up or up-down) until lift reaches to next floor.

Now the condition is checked again if the lift is reached at desired destination or not. If not the direction LEDs again starts blinking until life reaches to the next floor.

The condition of current floor with destination floor is being checked again and again.

As lift movers from one floor to another position LEDs set HIGH for the current floor and previous floor position LED is set LOW.

That way we can implement the lift elevator by just changing the LED pattern ON/OFF.

**Conclusion:** We have successfully demonstrated the lift elevator system using Raspberry Pi.

**Code:**

import time

import RPi.GPIO as GPIO

RUNNING = True

HIGH = 1

LOW = 0

#Position LEDs

FL0\_LED\_POS = 10

FL1\_LED\_POS = 12

FL2\_LED\_POS = 13

FL3\_LED\_POS = 16

#CALL Position LEDs

FL0\_LED\_CALL = 6

FL1\_LED\_CALL = 7

FL2\_LED\_CALL = 8

FL3\_LED\_CALL = 9

#CALL Switch

FL0\_SW = 17

FL1\_SW = 18

FL2\_SW = 19

FL3\_SW = 20

#DIRECTION LED

LED\_D0 = 21

LED\_D1 = 22

LED\_D2 = 23

LED\_D3 = 24

LED\_D4 = 25

LED\_D5 = 26

LED\_D6 = 27

NO\_OF\_FLOORS = 4 # No of floors for Lift Simulation Operation

NO\_OF\_DIR\_LEDS = 7 # No of LEDs used for the lift direction (on Board)

DEFAULT\_LIFT\_POS = 0 # The floor no where lift is positioned when program is executed

DIR\_LED = [ LED\_D0,LED\_D1,LED\_D2,LED\_D3,LED\_D4,LED\_D5,LED\_D6]

FLOOR\_POS\_LED = [FL0\_LED\_POS,FL1\_LED\_POS,FL2\_LED\_POS,FL3\_LED\_POS]

FLOOR\_CALL\_LED =[FL0\_LED\_CALL,FL1\_LED\_CALL,FL2\_LED\_CALL,FL3\_LED\_CALL]

FLOOR\_SW =[FL0\_SW,FL1\_SW,FL2\_SW,FL3\_SW]

def InitElevator():

GPIO.setmode(GPIO.BCM)

GPIO.setup(FL0\_LED\_POS,GPIO.OUT)

GPIO.setup(FL1\_LED\_POS,GPIO.OUT)

GPIO.setup(FL2\_LED\_POS,GPIO.OUT)

GPIO.setup(FL3\_LED\_POS,GPIO.OUT)

GPIO.setup(FL0\_LED\_CALL,GPIO.OUT)

GPIO.setup(FL1\_LED\_CALL,GPIO.OUT)

GPIO.setup(FL2\_LED\_CALL,GPIO.OUT)

GPIO.setup(FL3\_LED\_CALL,GPIO.OUT)

GPIO.setup(LED\_D0,GPIO.OUT)

GPIO.setup(LED\_D1,GPIO.OUT)

GPIO.setup(LED\_D2,GPIO.OUT)

GPIO.setup(LED\_D3,GPIO.OUT)

GPIO.setup(LED\_D4,GPIO.OUT)

GPIO.setup(LED\_D5,GPIO.OUT)

GPIO.setup(LED\_D6,GPIO.OUT)

GPIO.setup(FL0\_SW,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(FL1\_SW,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(FL2\_SW,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(FL3\_SW,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

for i in range(0,4):

GPIO.output(FLOOR\_POS\_LED[i],LOW)

for i in range(0,4):

GPIO.output(FLOOR\_CALL\_LED[i],LOW)

for i in range(0,7):

GPIO.output(DIR\_LED[i],LOW)

return

def GoingUP():

for i in range(0,7):

GPIO.output(DIR\_LED[i],HIGH)

time.sleep(0.5)

for i in range(0,7):

GPIO.output(DIR\_LED[i],LOW)

return

def GoingDOWN():

for i in range(0,7):

GPIO.output(DIR\_LED[6-i],HIGH)

time.sleep(0.5)

for i in range(0,7):

GPIO.output(DIR\_LED[i],LOW)

return

def GetFloorCall():

while True:

for call\_sw in range(0,4):

input\_state = GPIO.input(FLOOR\_SW[call\_sw])

time.sleep(0.1)

if input\_state == 0:

return call\_sw

try:

print "\nLift Operation Simulation using Python\n"

print "-----------------------------------------------\n"

InitElevator()

cur\_flr = DEFAULT\_LIFT\_POS # Variable for current lift floor (initially 0)

while RUNNING:

GPIO.output(FLOOR\_POS\_LED[cur\_flr],HIGH)

new\_flr = GetFloorCall()

if new\_flr > cur\_flr: # if (new floor > current floor) means lift is called to upper floor

tmp = cur\_flr # store current floor no into tmp variable

GPIO.output(FLOOR\_CALL\_LED[new\_flr],HIGH)

print "LIFT going UP to floor #%d" %new\_flr # print destination floor

while (tmp != new\_flr): # Use tmp value (incremental); till it becomes destination

GoingUP() # Glow direction LEDs in upward direction

GPIO.output(FLOOR\_POS\_LED[tmp],LOW)

tmp += 1 # Increment tmp value by 1

GPIO.output(FLOOR\_POS\_LED[tmp],HIGH)

time.sleep(0.5) # Sleep for 0.5 second (500 ms)

elif new\_flr < cur\_flr: # if (new floor < current floor) means lift is called to lower floor

tmp = cur\_flr # store current floor no into tmp variable

GPIO.output(FLOOR\_CALL\_LED[new\_flr],HIGH)

GPIO.output(FLOOR\_CALL\_LED[cur\_flr],LOW)

print "LIFT going DOWN to floor #%d" %new\_flr # print destination floor

while (tmp != new\_flr): # Use tmp value (decremental); till it becomes destination

GoingDOWN() # Glow direction LEDs in downward direction

time.sleep(0.01) # Sleep for 10 ms

GPIO.output(FLOOR\_POS\_LED[tmp],LOW)

tmp -= 1 # Decrement tmp value by 1

GPIO.output(FLOOR\_POS\_LED[tmp],HIGH)

time.sleep(0.5) # sleep for 0.5 second (500 ms)

cur\_flr = new\_flr # Once lift reaches the destination; current floor points to destination floor no

GPIO.output(FLOOR\_CALL\_LED[cur\_flr],LOW)

time.sleep(0.1) # Sleep for 1 second

# If CTRL+C is pressed the main loop is broken

except KeyboardInterrupt:

RUNNING = False

print "\Stopping Elevator"

# Actions under 'finally' will always be called

finally:

# Stop and finish cleanly so the pins

# are available to be used again

GPIO.cleanup()